Élaboration d'une interface tangible pour l'assemblage en CAO

Ludovic Garreau

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES
ET D'INFORMATIQUE

Par **Ludovic GARREAU**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPECIALITÉ : INFORMATIQUE

Élaboration d'une interface tangible pour l'assemblage en CAO

Soutenue le : 12 Septembre 2005

Après avis des rapporteurs :

Dominique Bechmann (LSIIT, Strasbourg)

Sabine Coquillart (INRIA, Grenoble)

Devant la commission d'examen composée de :

Pascal Weil (LaBRI, Bordeaux, Président)

Dominique Bechmann (LSIIT, Strasbourg, Rapporteur)

Sabine Coquillart (INRIA, Grenoble, Rapporteur)

Pascal Guitton (LaBRI, Bordeaux, Directeur de thèse)

Jérémy Legardeur (ESTIA, Bidart, Examinateur)

Jean-Claude Léon (L3S, INPG, Grenoble, Examinateur)

Nadine Rouillon-Couture (ESTIA, Bidart, Co-directrice de thèse)

Table des matières

Table des figures

Introduction

Sommaire

Contexte : l'assemblage en CAO

La CAO est une discipline ayant recours aux techniques informatiques pour créer un objet, en achever la forme et générer les données nécessaires à sa fabrication. De nombreux domaines (la mécanique, l'aéronautique, l'architecture) font appel à la conception assistée par ordinateur.
Depuis les années 1960, de nombreuses recherches portent sur les étapes de la fabrication afin d'optimiser la conception et le traditionnel triptyque coût, qualité et délais d'un produit ([Tic94]). C'est ainsi que sont apparus, entre autres, les services méthode chargés de faciliter les échanges entre le bureau d'études et la fabrication. Mais le réel besoin se situe au niveau de la phase de conception afin d'anticiper au plus tôt l'ensemble des contraintes du produit. Dans le domaine de l'assemblage mécanique, les logiciels CAO, tout en aidant le concepteur dans sa tâche, masquent des problèmes n'apparaissant que lors de la phase de fabrication. Par exemple, les difficultés de mise en position relative de deux pièces avant fixation ou encore les difficultés d'insertion d'une pièce par rapport aux autres telles que l'inaccessibilité ou les collisions. En effet, lors de la phase CAO les concepteurs en bureau d'études ne sont jamais confrontés à la réalité du montage.
Le point de départ de cette thèse est de prendre en compte des contraintes réelles le plus tôt possible dans le processus de conception, et de mener ainsi une réflexion plus accrue sur la phase d'assemblage. Pour cela, nous proposons un dispositif d'interaction différent de la souris classiquement utilisée avec les logiciels de CAO.

Cadre : l'interaction

Le processus d'interaction peut être vu comme un système bilatéral ayant à une extrémité la donnée numérique et à l'autre extrémité l'utilisateur. La donnée se situe dans l'espace virtuel alors que l'utilisateur est dans l'espace réel. Pour faire communiquer ces deux éléments, nous considérons trois composantes principales : d'**action**, de **perception** et de **métaphores** d'interaction. Nous regroupons sous le terme de composantes d'**action** tous les périphériques d'entrée tels que la souris, la clavier, la tablette graphique, qui permettent à l'utilisateur de transmettre des informations au système. Les composantes de **perception** correspondent aux périphériques de sortie et permettent de communiquer à l'utilisateur des informations sur une donnée ou une action. Ce sont, entre autres, les systèmes d'affichage, les imprimantes, les systèmes haptiques ou les hauts parleurs.
Notons qu'un processus d'interaction peut faire intervenir plusieurs composantes d'action et de perception et que certains objets peuvent faire partie des deux catégories. Par exemple un périphérique de perception tel un écran peut être utilisé avec un système haptique qui est à la fois un périphérique d'action et de perception.
Dans le sens commun une **métaphore** est une image qui réfère à un modèle, c'est à dire à une structure avec des liens qui sont déjà assimilés et compris dans le quotidien ou à une structure qui fait appel à une compréhension globale d'un phénomène. La métaphore d'interaction permet de suggérer le processus d'interaction à l'utilisateur. Par exemple, dans WIM [SCP95] (World In Miniature) l'utilisateur modifie le point de vue d'une scène virtuelle par l'intermédiaire d'une maquette miniature numérique. Cette maquette et les actions qu'elle inspire constitue une métaphore d'interaction de la manipulation de la scène.

Depuis les premiers systèmes d'interaction, comme le Sketchpad [Sut63] de Sutherland en 1963, la communauté IHM cherche et invente de nouveaux périphériques de perception et d'action répondant à de nouveaux besoins. Les premiers périphériques d'action, stylo optique ou souris, possédaient deux degrés de liberté et ne permettaient pas de retour d'information de la part du système. Aujourd'hui, il existe des périphériques disposant de plus de degrés de libertés – la spaceMouse [Spab] en compte six – des périphériques haptiques – les systèmes spidar [MBL$^+$04] – et des systèmes de capture de mouvement [dlRG03].
De même, de nouveaux périphériques de perception sont apparus comme une alternative aux écrans classiques. Certains travaux cherchent à appréhender l'interaction en stimulant plus de sens chez l'utilisateur ([NC96], [CNS93]), d'autres immergent l'utilisateur avec des systèmes de casque virtuel ([PPS99]) ou des plans de travail ([LSR$^+$00]).
Aujourd'hui, il existe un large panel de techniques d'interaction. Elles vont de la simple ligne de commande aux interfaces de types WIMP (Window Icon Menu Pointing device) en passant par les interfaces 3D (par exemple : le navigateur de SGI [3Dn02], le gestionnaire de fenêtre 3D de Karlsson [Kar02]). Toutes combinent

l'utilisation des composantes d'action, de perception et de métaphores d'interaction.

Motivation : l'interaction au moyen d'interfaces tangibles

Les techniques d'interaction ne semblent pas adaptées pour confronter le concepteur aux contraintes réelles de l'assemblage. La principale raison est l'écart entre les composantes d'action existantes et les données virtuelles associées que ce soit au niveau de la forme, de la couleur ou du comportement. Ainsi, un système de Réalité Virtuelle utilisant des gants de données pour sélectionner, manipuler et assembler des objets numériques est capable de rendre le contact entre deux d'entre elles uniquement par une composante de perception visuelle. La même application basée sur des phantom [Che99a] ne représentera pas directement les déplacements dans l'espace et donc la composante de perception liée au comportement.
Même si cet écart est en partie comblé par les métaphores d'interaction, nous pensons que pour une tâche d'assemblage les composantes d'action doivent posséder une partie réelle plus significative (forme, couleur, ou comportement) de la donnée manipulée. En effet, l'utilisateur ne sera jamais aussi bien confronté aux contraintes d'assemblage qu'en simulant le processus dans la réalité.
Aish [R.79] et Frazer [SSWF00] ont proposé une technique d'interaction s'appuyant sur cette idée : les interfaces tangibles. Les interfaces tangibles (du latin tangere : toucher) utilisent des objets du monde réel pour contrôler et symboliser les données numériques et en permettre une appréhension plus directe. En effet, l'utilisateur a déjà une connaissance de l'objet réel tant au niveau de ce qu'il symbolise que de la façon de l'utiliser.
Ces objets ont la particularité de rassembler les composants de perception visuelle, haptique et du comportement ainsi que les composantes d'action. De ce fait, la conception des objets physiques dépend fortement des données numériques manipulées et des actions réalisables. Ainsi, chaque interface tangible est créée en fonction d'un domaine d'application déterminé.

Au cours de cette thèse, nous avons conçu et réalisé une interface tangible, nommée SKUA, dédiée à l'activité d'assemblage de pièces mécaniques en phase de conception. Notre interface tangible ne remplace pas la souris et le clavier mais les complète en procurant une représentation dans le monde réel.

Champ d'application : le projet ESKUA

Durant l'année 2000, un projet nommé ESKUA a vu le jour. ESKUA, acronyme d'Expérimentation d'un Système Kinésique Utilisable pour l'Assemblage, est le nom du projet d'expérimentation de l'interface tangible SKUA. SKUA est

le prototype d'interface tangible permettant la manipulation, l'assemblage et la visualisation de modèles numériques 3D qui a été développé lors de cette thèse. L'idée initiale du projet ESKUA (octobre 2000) vient de Nadine Couture (enseignant chercheur au LIPSI de l'ESTIA). Lors d'un cours à ERGO-IHM 2000 sur la multimodalité, Laurence Nigay (maître de conférence, université Joseph Fourier de Grenoble) présente un système permettant de manipuler des fichiers informatiques avec des cubes en bois : les mediaBlocks ([UIG98]) d'Ihsii et Ullmer. En 2001, Nadine Couture et Pascal Guitton (professeur, LaBRI-Bordeaux 1) proposent de co-encadrer une thèse visant à étudier le concept d'interface tangible et la possibilité de réaliser une interface tangible adaptée à la manipulation et à l'assemblage en CAO. Fin de l'année 2002, Jeremy Legardeur (enseignant chercheur, mécanique) rejoint l'équipe en enrichissant le projet de compétences propres à l'assemblage mécanique. À la fin de l'année 2003, la région Aquitaine apporte son concours financier au projet. Début 2004, Fabrice Depaulis (post doctorant, informatique) travaille sur la formalisation de la partie théorique des interfaces tangibles et leur validation. Aujourd'hui, fin 2004, cinq personnes travaillent sur le projet ESKUA : Nadine Couture (enseignant chercheur, informatique), Fabrice Depaulis (post doctorant, informatique), Ludovic Garreau (doctorant, informatique), Jérémy Legardeur (enseignant chercheur, mécanique). Ce projet a déjà fait l'objet de cinq publications internationales ([GLC03],[LGC04],[LGC03],[GC03], [DCGL05]).

Organisation du mémoire

Dans cette thèse, nous nous intéressons à la réalisation d'une interface tangible adaptée aux contraintes d'assemblage et plus particulièrement à la partie réelle de cette interface.

Dans le **premier** chapitre nous étudions l'Interaction Homme Machine (IHM) au travers des facteurs les plus déterminants pour la création d'Interfaces d'Interaction Homme-Données (I2HD), puis nous introduisons et décrivons les interfaces utilisateur tangibles (TUI).

Dans le **second** chapitre, nous étudions les différentes Interfaces Interaction Hommes-Données (I2HD) conçues pour la manipulation géométrique (rotation, translation). De cette étude, nous proposons une taxinomie présentant les critères nous semblant les plus importants pour la réalisation d'une I2HD centrée sur la manipulation géométrique de modèles 3D. Dans le chapitre **3**, nous analysons la tâche liée à notre interface : l'assemblage. Ensuite, en fonction des caractéristiques de l'assemblage, de celles d'une interface tangible et de notre taxinomie nous réalisons la conception détaillée des trois composantes de SKUA : les inter-

acteurs [GLC03], la plate-forme [GC03], la communication interacteur-ordinateur. Au cours du chapitre **4**, nous présentons la concrétisation de nos travaux avec la description des prototypes. Puis nous dressons une première évaluation [LGC04] de notre travail en analysant la perception des interacteurs et le pressentiment de l'utilisateur vis à vis des interacteurs. Dans le **dernier** chapitre nous récapitulons notre contribution et nos perspectives de travail.

Chapitre 1

Interaction Homme Machine et Interfaces Utilisateur Tangibles

Sommaire

Dans la première partie de ce chapitre nous décrivons l'Interaction Homme Machine (IHM). Notre but n'est pas de pratiquer une étude rigoureuse et exhaustive, mais de réaliser une introduction pour définir quelques concepts élémentaires, dont les facteurs les plus déterminants pour la création d'Interfaces d'Interaction Homme-Données (I2HD). Sous ce vocable, nous rassemblons tous les éléments permettant l'interaction entre l'utilisateur et la donnée virtuelle : les périphériques d'entrée, les périphériques de sortie et l'interface logicielle. Nous utilisons le terme

I2HD pour insister sur l'importance de la donnée dans le processus d'interaction. Dans la deuxième partie, nous introduisons puis décrivons les interfaces utilisateur tangibles (TUI) qui sont le type d'I2HD que nous avons étudié et créé au cours de cette thèse.

1.1 Interaction Homme Machine

Dans cette thèse, nous définissons l'interaction comme le dialogue homme-machine permettant une action réciproque entre l'utilisateur d'une machine exécutant un programme et l'exécution de ce programme. Elle est souvent représentée par le triplet <utilisateur, tâche, machine> ([The01]). L'utilisateur se trouve généralement dans l'univers réel pour exécuter une tâche dans le domaine virtuel. On peut remarquer que l'ordinateur qui est le lien entre le réel et le virtuel est aussi, paradoxalement, la frontière (cf. fig 1.1). L'IHM en tant que discipline est apparue au début des années 80 dans le but d'étudier le processus d'interaction.

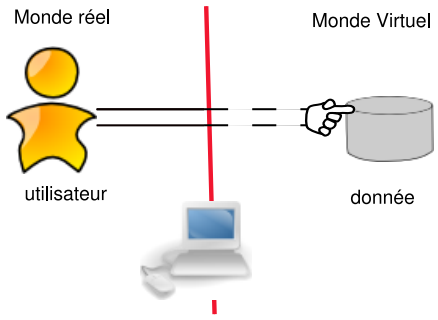


FIG. 1.1 – Simple illustration de l'interaction.

Elle est au confluent l'agrégation de plusieurs sciences telles l'informatique, la psychologie et les sciences cognitives (cf fig. 1.2). Une des premières revues et conférences dans ce domaine est ACM SIGGRAPH en 1972, qui se spécialise dans l'IHM en 1982 avec la création de SIGCHI, et HCI (Human Computer Interaction) en 1985. ACM ([HBC+96]) a proposé de formaliser l'«IHM» en 1992 :

« L'Interaction Homme-Machine est une discipline qui concerne la conception, l'évaluation et l'implémentation de systèmes informatiques interactifs pour des utilisations humaines et l'étude des phénomènes principaux les entourant ».

1.1.1 IHM : un conglomérat de sciences

La recherche sur l'interaction est reconnue comme reliée à plusieurs sciences ([MF97], [Car97]) allant de la psychologie à la conception graphique en passant par l'étude du comportement (cf. fig. 1.2).

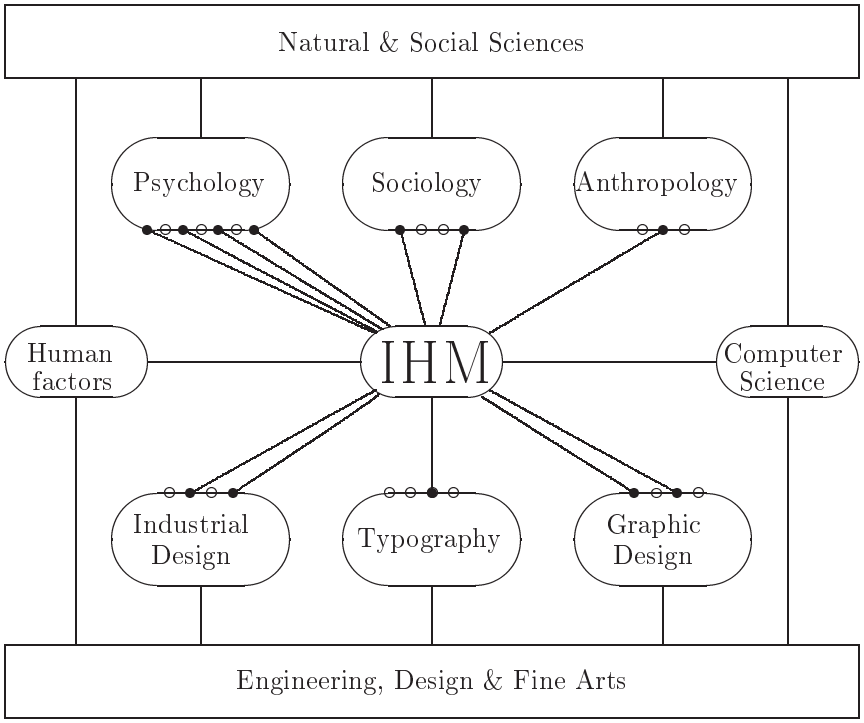


FIG. 1.2 – Schéma représentant l'interférence des spécialités en IHM (reproduit d'après [MF97]).

Pour concevoir des modèles d'IHM, Mosley et Smith ([MS86]) soulèvent les difficultés des ingénieurs en IHM à traduire les directives généralement exprimées en règles de conception et contraintes liées à ces domaines. L'IHM a fait évoluer la conception d'I2HD en intégrant l'utilisateur dans le processus d'interaction. Cette évolution est le passage d'une conception centrée sur la tâche à une conception humanocentrique[1] dans le but de mieux répondre aux besoins de l'utilisateur. Il existe des modèles de conception permettant de modéliser et de décrire l'exécution humaine d'une tâche. Pour plus de détails le lecteur pourra se reporter à l'étude des techniques "Star Life cycle"([HH93]) ou GOMS ([CMN83]). La conception humanocentrique d'I2HD ajoute une étude supplémentaire dans la conception : "l'utilisateur". Nous devons étudier sa culture et les codes spécifiques à son travail. En effet, ces éléments font partie intégrante du processus d'interaction.
La conception d'un modèle d'interaction va au-delà des contraintes informatiques.

[1]Se dit de ce qui est axé sur les besoins de l'humain et non de la machine, c'est-à-dire sur l'interface la plus naturelle possible pour l'utilisateur [OLF].

Il convient d'étudier chaque composante de l'interaction "utilisateur, tâche, interface", en considérant leurs relations d'interdépendance.

Dans le paragraphe 1.1.2, nous décrivons l'influence et le rôle de l'interface dans le processus de communication. Dans la section 1.2, nous décrivons le principe d'I2HD qui fut notre point de départ durant cette thèse : l'Interface Utilisateur Tangible. Dans la section 1.2, nous décrivons les Interfaces Utilisateur Tangibles, point de départ de la conception de l'I2HD réalisée au cours de cette thèse.

1.1.2 Le processus de communication

Les interfaces logicielle et matérielle sont au cœur du processus de communication. Elles induisent les actions de l'utilisateur et traduisent les réactions du système (cf fig. 1.3).

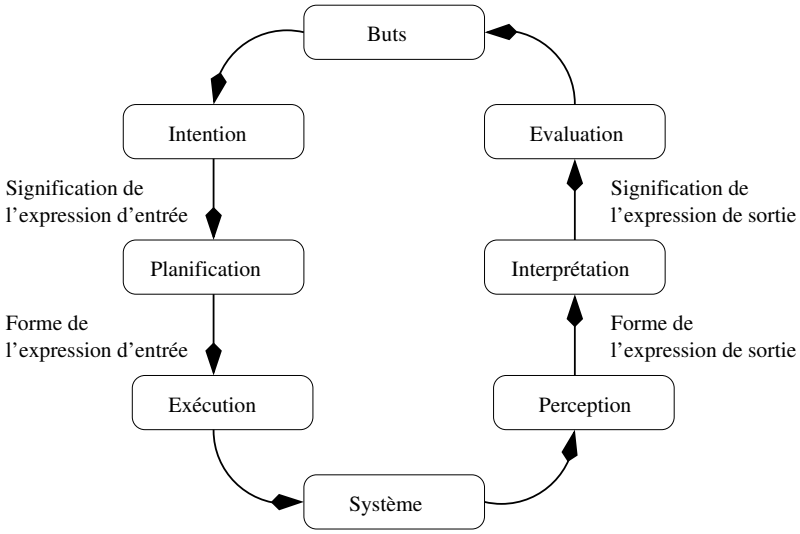


FIG. 1.3 – Théorie de l'action de Norman (extrait de [Pat99]).

La difficulté dans la réalisation d'I2HD est de réduire la distance entre l'utilisateur et la donnée. Il ne s'agit pas d'une distance métrique mais de l'écart entre la tâche virtuelle (par ex : ouvrir un fichier) et la tâche réelle (par ex : cliquer sur l'icône "ouvrir"). Norman parle de fossés d'exécution et d'évaluation :

- Fossé de l'exécution : spécifier des intentions, choisir une séquence d'actions, exécuter les actions en respectant les formats offerts par le système.
- Fossé de l'évaluation : percevoir l'état du système, l'interpréter et comparer l'intention au but final.

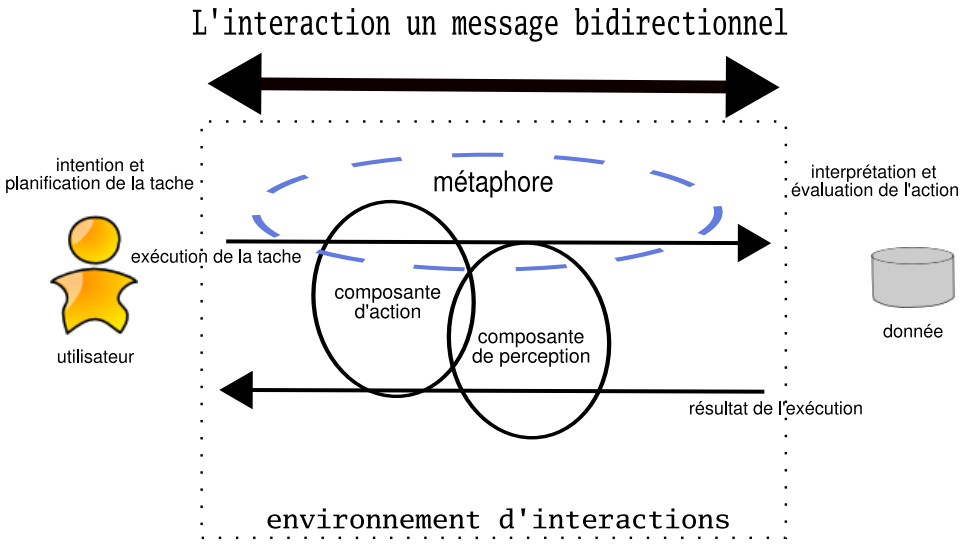


FIG. 1.4 – Schéma du processus d'interaction.

Ces deux fossés représentent la distance entre le geste réalisé et l'action souhaitée. Le but de toute interface est de réduire ces fossés en modifiant la métaphore d'interaction. Par exemple, Fuchs [FMP01] cherche à réaliser des interfaces comportementales ou «*le dispositif matériel exploite un comportement humain, naturel et n'implique pas une période d'apprentissage importante et complexe pour pouvoir être utilisé*».

Dans l'IHM, l'utilisateur modifie la donnée en communiquant avec l'ordinateur via les composantes d'action et de perception. Les composantes d'action sont classiquement les interfaces matérielle (souris, clavier) et logicielle (menu, icône). La combinaison de ces deux interfaces (cf fig. 1.4) fournit l'ensemble des actions réalisables par l'utilisateur et compréhensibles par l'application.
L'interface est cohérente dans la mesure où les actions possibles sont perceptibles par l'utilisateur et sont réalisables au travers de métaphores efficaces[2]. Cette cohérence est dépendante des standards informatiques mais également de la culture et des connaissances de l'utilisateur.
Dans le but de permettre une compréhension rapide de l'interaction, l'interface doit faire référence aux acquis de l'utilisateur aux travers des métaphores d'interaction. Ainsi, nous utilisons les standards informatiques : par exemple, le menu «fichier» contient les rubriques «enregistrer» et «ouvrir», l'icône de la disquette signifie «enregistrer». Ceci permet d'appréhender l'interface sans utilisation préalable. Nous pouvons utiliser des codes spécifiques à la tâche en plus des standards informatiques. Par exemple, dans un logiciel de modélisation, l'icône "sphère en

[2] Nous entendons par métaphores efficaces, des métaphores qui permettent à l'utilisateur de réaliser la tâche qu'ils souhaitent rapidement.

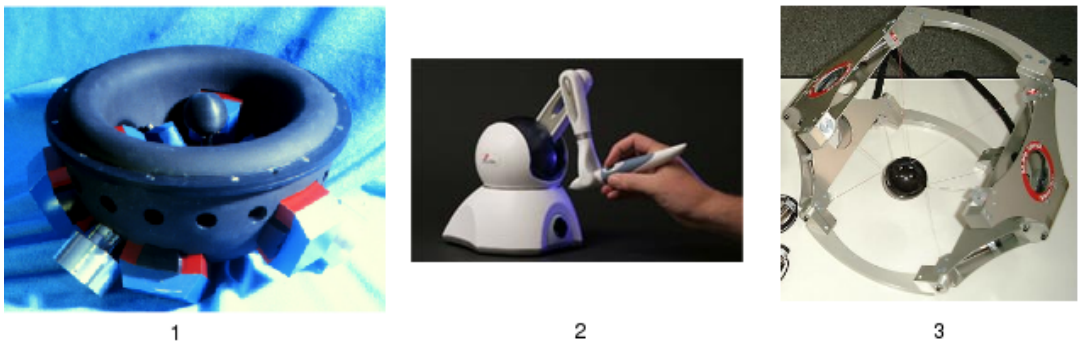


FIG. 1.5 – Photographie de différents systèmes haptiques.

fil de fer" génère le rendu en fil de fer. Cet icône logique pour les spécialistes en informatique graphique, est incompréhensible pour un utilisateur lambda car l'icône fait référence à la connaissance du domaine de la tâche, la modélisation 3D.
Les interfaces logicielles et matérielles peuvent également exploiter des comportements humains dans le but de simplifier l'interaction. Fuchs désigne l'interface matérielle utilisant un comportement acquis[3] par le terme de «schème».
Pour tendre vers une interface comportementale, il faut rapprocher l'intention de l'exécution en introduisant des schèmes dans l'interface matérielle. Il existe deux manières de concevoir une telle interface. La première est dans la création de la forme du périphérique. Par exemple, le périphérique peut avoir la forme d'un stylo si l'objet virtuel manipulé est un stylo. La seconde est dans la manipulation du périphérique. L'objet virtuel et le périphérique ont le même comportement : par exemple, une rotation du périphérique produit une rotation de la donnée.

Un autre moyen d'améliorer l'interaction est de permettre à l'utilisateur de "sentir" la donnée.
Hong Z. Tan [Tan00] et Lécuyer [LCea] soulignent l'apport d'une interface haptique dans le processus d'interaction. L'utilisation d'un système haptique procure à l'utilisateur les sensations de contacts entre la composante d'action et la donnée manipulée. Dans ce cas, la composante d'action est également une composante de perception. Il existe plusieurs approches pour restituer une sensation de retour haptique tels l'utilisation de périphériques à retour haptique et l'utilisation d'ustensiles ("props"). Parmi les systèmes à retour haptique, les plus connus sont : mécanique à retour d'effort [PHA] (2, fig. 1.5), magnétique [BH97] (1, fig. 1.5), dispositif à câbles [TCH+03] (3, fig. 1.5).

Les systèmes basés sur les forces magnétiques de Lorenz tels les Maglev 1 et 2 sont assez rares et nécessitent des dispositifs de taille importante. Les systèmes mécaniques, comme les joystiks à retour d'effort (Logitech Wingman Force) ou le

[3] acquis : un comportement ne nécessitant pas d'effort cognitif important.

Phantom [Che99b] sont des périphériques qui sont commercialisés. Ils s'utilisent, respectivement, avec des jeux vidéo et des logiciels CAO (FreeForm). Les sensations avec les joysticks sont perfectibles et le Phantom reste encore assez cher. Dans le laboratoire Sato-Koike au Japon [Gro], un système à base de câbles, le SPIDAR [MBL+04], a été réalisé. Chaque câble est motorisé et permet d'exercer un retour d'effort dans une direction. Ce système permet une souplesse d'utilisation plus grande, car il peut être utilisé en position assise avec un écran classique ou debout avec un grand écran. L'avantage de cette technique est de permettre de positionner les efforts avec plus de souplesse par l'intermédiaire des câbles. Tous les systèmes sont performants pour simuler la présence de la donnée dans le monde réel en produisant une résistance active.
Tous ces systèmes cherchent à palier l'absence de la donnée dans le monde réel. Comme Mark, Randolph et Finch [MRF+96] nous pensons que ses systèmes sont difficiles à réaliser à cause de la fréquence d'acquisition et de la gestion de la résistance.

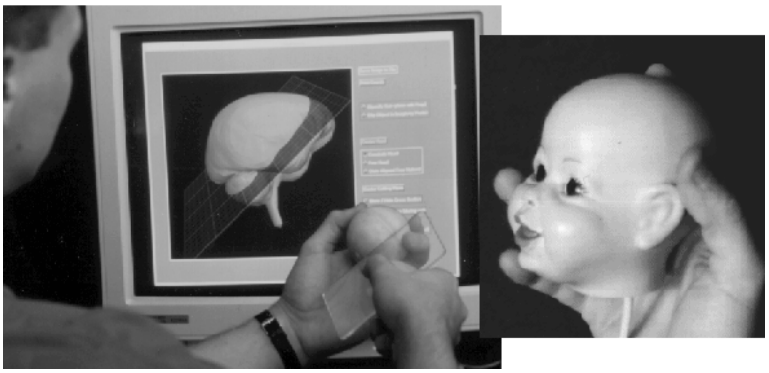


FIG. 1.6 – Photographie des accessoires inventé par Hinckley.(extrait de [HPGK94])

Dans d'autres travaux, les chercheurs proposent des accessoires ([SCP95], props) à l'utilisateur pour donner la sensation de présence dans le monde réel. Dans ce cas nous parlons d'"haptique passif". Lindeman [LTSC02] souligne l'importance d'un retour haptique même passif et utilise, par exemple, une palette dans le monde réel [LST01] comme support pour des sliders virtuels dans le but de faciliter leurs accès à l'utilisateur. Dans certains cas, l'accessoire peut également servir de métaphore à la donnée et de par sa forme il peut induire à l'utilisateur son comportement. Par exemple, Hinckley [HPGK94] a créé une application basée sur cette idée pour visualiser de coupe d'un cerveau (fig. 1.6). L'utilisateur se sert d'une tête de poupée pour manipuler un cerveau et déplace une tablette relativement à la tête pour préciser la position de la coupe. De même, Fjeld [FSSK02]

montre l'efficacité d'une interface tangible pour représenter la position des données dans l'espace.
Plus récemment, Tang, Owen, Biocca et Mou [TOBM02], [TOBM03] ont montré l'avantage d'utiliser des accessoires pour l'assemblage avec dans système de réalité augmentée pour les tâches d'assemblage. Contrairement à nos travaux, ils n'ont pas travaillé sur la réalisation d'accessoires adaptés à la tâche d'assemblage, ils ont utilisé de simples Lego^{TM}. Isidoro et Sclaroff [IS98] proposent des objets mou réels, les "Active Voodoo Dolls", pour déformer les objets virtuels (fig. 1.7).

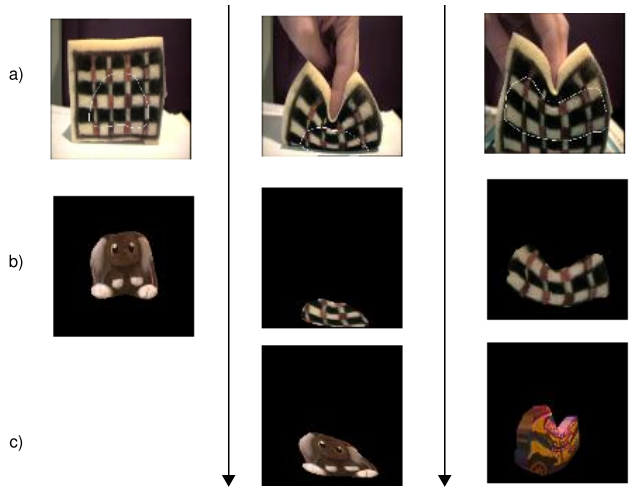


FIG. 1.7 – Photographie des active Voodoo Dolls.(extrait de [IS98])

Précédemment, nous avons présenté des accessoires pouvant être une métaphore de la donnée par sa forme, une métaphore du comportement de l'objet virtuel ou encore une métaphore de la forme de la donnée. Ainsi, la conception d'accessoires adaptés est une des principales difficultés pour la création d'I2HD à cause de leur variété et de leur fonction. Pierce, Stearns et Pausch [PPS99], reconnaissent que l'utilisation d'accessoires est pertinente pour la manipulation d'objet, mais ils expliquent qu'il est très difficile de fournir un accessoire générique représentant tous les objets virtuels. D'ailleurs dans leur application, les Voodoo Dolls, l'utilisateur équipé de gants manipule une copie numérique des objets et non des accessoires du monde réel.

Dans la suite nous décrivons les interfaces tangibles pouvant être considérées comme des systèmes haptiques passifs s'appuyant sur un lien sémantique plus important entre la composante d'action et la donnée.

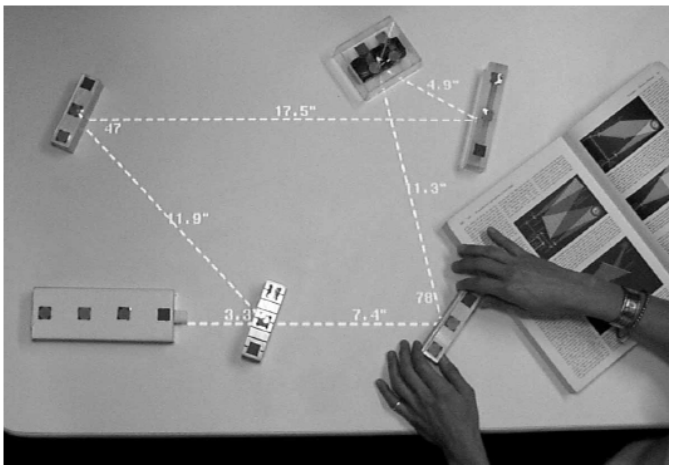


FIG. 1.8 – Photographie de l'application "Luminous Room"(extrait de [UUI99].

1.2 Interfaces Utilisateur Tangibles (TUI)

Après avoir évoqué l'interaction, nous décrivons en détail le principe de l'I2HD sur lequel nous avons fondé notre étude : les interfaces tangibles utilisateur. De la même manière que les "interfaces graphiques utilisateur" sont appelées "interfaces graphiques", nous appellerons par la suite les "interfaces tangibles utilisateur" "interface tangible" ou TUI. Elles sont également appelées "graspable user interface" ([FIB95], [ESB99]), "natural user interface" ([MP96]) ou "Tangible Query Interfaces" ([UIJ03]).

1.2.1 Présentation

Le fondement du concept ([Ull95]) des interfaces tangibles est de donner une forme physique figurative (objet tangible) à une donnée numérique dans le but de pouvoir la manipuler. Nous entendons par "figurative" que l'objet tangible représente la donnée numérique ou symbolise une tâche de par sa forme, sa couleur, sa position. Une souris sortie de son contexte informatique ne génère aucune information. En revanche, les objets réels symboliques permettent d'appréhender la manipulation. Par exemple, une maquette de maison peut être utilisée pour symboliser la manipulation de tous les types de bâtiments. Dans l'application "Luminous Room" ([UUI99]), les objets réels sont des imitations de prismes et de miroirs qui servent à dévier un rayon lumineux calculé par l'ordinateur. Dans ce cas, l'objet réel et l'action virtuelle sont en concordance, si bien que même sans ordinateur l'utilisateur peut comprendre son action.

1.2.1.1 Historique

Sharlin, Watson, Sutphen et Al. ([SWS+01]), Ullmer et Ishii ([UI01]), et Anderson, Frankel, Marks et Al. ([AFM+00]) s'accordent pour dire que les premières interfaces tangibles furent créées à la fin des années 70 par Frazer ([SSWF00]) et Aish ([Ais79]). L'interface tangible d'Aish permet à l'utilisateur de manipuler la structure d'un bâtiment et d'en calculer les propriétés thermiques. L'utilisateur insère des panneaux rectangulaires sur un socle. Chaque panneau symbolise un pan de mur. Le socle symbolise le sol. Il est relié à l'ordinateur pour communiquer les actions de l'utilisateur (nous y reviendrons plus en détails dans le paragraphe 2.2.7). L'utilisation reste simple même pour une personne inexpérimentée. L'idée de déplacer l'action dans l'espace réel avec des objets physiques est à l'origine des interfaces tangibles. Les interfaces tangibles ont été formalisées par Ullmer et Ishii dans [Ull95] puis dans [Ull02]. L'explication et la description des TUI dans cette thèse s'appuient fortement sur les travaux d'Ullmer et Ishii ([UI97a], [UUI99], [UI97b], [GOH98]) mais aussi sur les travaux d'autres chercheurs comme Anderson ([AFM+00]) et Fjeld ([FBR86]).

1.2.1.2 Concept

Dans la figure 1.1 page 8, nous avons illustré le fait que les interfaces logicielles et matérielles sont les parties à traiter pour tendre vers une interaction directe. Une des bases du concept des TUI est de donner, dans le monde réel, des **représentations physiques** (cf. fig 1.10) aux données virtuelles contrairement aux interfaces classiques (cf. fig 1.9). Le schéma "Modèle/Vue/Controleur" de la figure 1.9 illustre l'ambiguïté du processus d'interaction classique, oscillant entre le domaine physique et le domaine numérique. Le modèle est dans le domaine physique tandis que le contrôle (souris et interface graphique) et le retour sont à la fois dans les deux domaines. En revanche, cette ambiguïté n'apparaît pas dans le schéma "Modèle/Controleur/Représentation Physique et digital" 1.10 où le contrôle est une représentation physique de la donnée. Dans ce cas, le processus d'interaction est entièrement dans le domaine réel. La partie tangible de l'interface est appelée *graspable* par Ernst, Schäfer et Bruns [ESB99]. Dans [UI01], Ullmer et Ishii référencent les termes *"Artifact, Container, Tangible Object, Phicons, Objects, props"*. Nous avons choisi d'utiliser le terme français "interacteur" pour désigner la partie tangible de l'interface. [4] Dans l'interaction nous définissons deux espaces : l'espace de perception et l'espace d'action. L'espace d'action est l'espace où l'utilisateur agit. L'espace de perception est l'espace où l'utilisateur observe le résultat de ses actions.

[4] Nous utilisons le néologisme *Interacteur*, composition des mots "interaction" et "acteur", car la partie tangible permet l'*interaction* entre les *acteurs* : la donnée et l'utilisateur. Le mot "interacteur" n'est ni référencé dans l'encyclopédie Wikipédia ([Wik]) ni dans les dictionnaires - Larousse et Hachette - de la langue française.

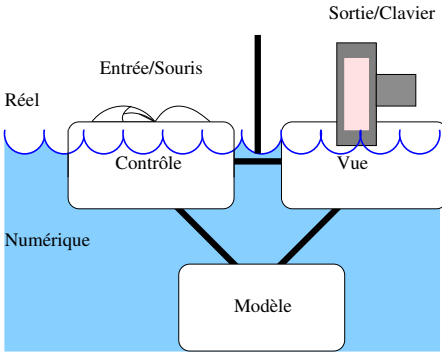


FIG. 1.9 – Représentation du MCV dans une GUI classique.
(reproduit d'après [UI00])

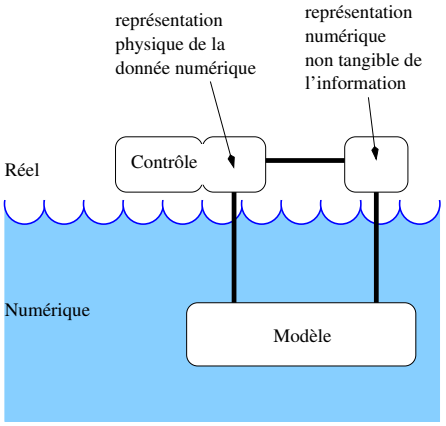


FIG. 1.10 – Représentation du MCRdp dans une TUI.
(reproduit d'après [UI00])

Les principales caractéristiques d'un interacteur, définies par Ullmer ([Ull02]), sont la manipulation physique, la représentation physique, la représentation du comportement et la configuration spatiale. Dans [SSWF00], Sutphen, Sharlin, Watson et Frazer caractérisent leur interface tangible en utilisant les termes d'affordance, de support épistémique et de pragmatique des actions et de «synchronisation». Par la suite, nous rapprochons les deux approches pour les unifier.

La **représentation du comportement** (**affordance**) signifie que la forme de l'interacteur évoque sa fonctionnalité par ses propriétés physiques. Il doit exis-

ter un lien d'action entre la donnée et l'interacteur, autrement dit, l'action désirée doit pouvoir être exécutée sur l'interacteur.

La **configuration spatiale** (**synchronisation**) de la perception concerne la correspondance entre l'espace d'action des mains et la position réelle des objets dans l'espace de perception. Les interfaces tangibles fonctionnent principalement sur des modes isotoniques : l'utilisateur applique réellement les déplacements et les rotations des interacteurs.

La **représentation physique** (**pragmatique**) suppose que la forme de l'interacteur fasse référence à l'objet virtuel où à la tâche. Cette notion est expliquée dans le paragraphe 1.2.1 sous le terme "forme figurative".

La **manipulation physique** (**épistémique**) représente une même notion complexe à présenter. Elle est comme la possibilité "d'épreuves et d'erreurs" que nous exécutons tout en essayant de progresser. Certaines de ces manoeuvres n'apporteront rien, tandis que d'autres indiqueront de nouvelles informations et des directions menant au but.

En recoupant ces concepts, nous avons extrait quatre règles de base pour guider la conception de l'interacteur en fonction de la tâche et de l'utilisateur :

Règle 1 *L'interacteur doit suggérer sa fonction par ses propriétés physiques.*

Règle 2 *L'espace d'action doit être confondu avec l'espace de perception.*

Règle 3 *L'interacteur doit avoir une forme proche de celle de l'objet numériquement manipulé.*

Règle 4 *L'interacteur doit être manipulable directement avec les mains et tangible.*

Une interface tangible possède une système faisant la liaison entre les interacteurs et l'ordinateur - comme le système optique d'une souris permet d'évaluer les déplacements - permettant de connaître à tout moment l'état[5] de chaque interacteur. Nous avons regrouper les systèmes en deux catégories : les systèmes avec des **interacteurs actifs** et ceux avec des **interacteurs passifs**. Les interacteurs actifs possèdent des composants électroniques permettant de connaître leurs états et de les communiquer à l'application. Les interacteurs passifs laissent un système extérieur évaluer leurs états et les communiquer à l'application.
Les deux parties suivantes décrivent des exemples d'interfaces tangibles avec des interacteurs actifs puis passifs.

[5] Les états d'un interacteur nécéssaires varient en fonction de l'I2HD, cela peut être la position, la forme, un bouton enclenché.

1.2.2 TUI avec des interacteurs actifs

Dans les systèmes avec des interacteurs actifs que nous avons étudié, ils s'agit souvent de répèrer les interacteurs par rapport à une base relié au PC dont la position est connue. Les interacteurs fixés à cette base, par l'utilisateur, communiquent leurs identifiants et leurs états. La principale difficulté est de créer un système électronique permettant de connaître l'état de l'interacteur et de le communiquer sans imposer de contraintes fortes sur la forme de l'interacteur et satisfaisant les quatre règles de conception précédemment citées.
Dans les paragraphes suivants, nous décrivons le travail du laboratoire MERL[6] et de l'université d'Osaka.

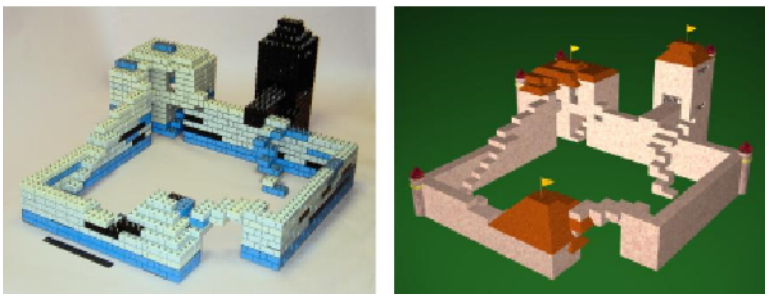


FIG. 1.11 – Projet du laboratoire Merl (extrait de [AFM+00]).

1.2.2.1 Utilisation de connexions

L'application ([AFM+00]) repose sur le principe d'assemblage d'éléments de type Lego^{TM} (cf fig. 1.11). Le but de cette application est de créer des modèles 3D de bâtiment de manière simple sans utiliser de modeleur. L'utilisation est intuitive. L'utilisateur assemble les interacteurs pour créer une structure symbolisant les murs du bâtiment (cf fig. 1.11) de la même manière qu'un maçon construit les murs d'une maison. Ensuite il relie la structure à un ordinateur pour générer le modèle 3D. Les textures, les toits et les fenêtres sont générés par un traitement logiciel.

Dans cette interface tangible, les interacteurs communiquent leur état à l'ordinateur. Chaque interacteur possède huit connecteurs sur le dessus et huit prises jacks en dessous (cf fig. 1.12), une prise pour l'alimentation et une pour les signaux bidirectionnels. Ensuite, un logiciel basé sur des règles d'architecture interprète les structures telles des constructions de type bâtiment. Il analyse la forme et la position de chaque interacteur et ajoute des détails, toits, textures, fenêtres, au modèle 3D.
Dans cette première application, nous voyons que le résultat est très proche de l'action. L'utilisateur peut travailler et entrevoir le résultat de ses actions par la

[6]MERL : Mitsubishi Electric Research Laboratory

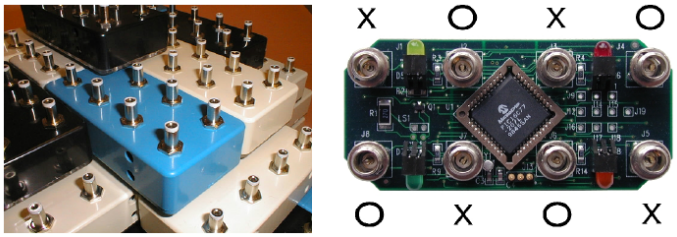


FIG. 1.12 – Un interacteur du modèle de Segal (extrait de[AFM+00].

simple vue des interacteurs, sans allumer l'ordinateur. La construction des murs est rapide et simple car l'utilisateur se sert de la seule fonctionnalité possible avec les interacteurs : l'assemblage. La contre-partie est la durée[7] du traitement logiciel pour le positionnement des éléments décoratifs.

FIG. 1.13 – Construction à partir des "ActiveCubes"(extrait de [SIW+02]).

1.2.2.2 Addition de capteurs aux interacteurs

ActiveCube [KIK01], [KIMK00] est une interface tangible pour construire une forme 3D (cf. fig. 1.13) et interagir avec elle. Les interacteurs sont des blocs de types Lego^{TM} munis de différents capteurs, par exemple : capteurs sonores et détecteurs d'obstacle. Il s'agit de cubes en plastique de 5 centimètres d'arêtes reliés entre eux par le biais de connecteurs mâle-femelles. L'ensemble des interacteurs connectés génère une structure physique. La topologie est reproduite sur l'écran de l'ordinateur. Pour débuter une construction, l'utilisateur se sert d'un cube spécifique relié au PC par une liaison filaire. Les autres cubes possèdent un microprocesseur pour communiquer leur identifiant, les faces connectées et les données du capteur. Reliés entre eux, les cubes forment un réseau local. Le cube spécifique établit le lien entre le réseau formé et le PC. Cette application

[7] Dans le papier [AFM+00] datant de 2000, le temps nécessaire est d'une heure.

diffère de la précédente par l'utilisation de capteurs (luminosité, obstacle, son) qui augmentent les possibilités d'actions. Par exemple, si l'utilisateur approche sa main près d'un détecteur d'obstacles un son est produit. Le volume sonore est proportionnel à la distance entre la main et le détecteur. Une série de test orientée manipulation avec un panel d'utilisateur réalisé par Kitamura, Itoh et Kishino [KIK01] ont prouvé que les ActiveCube était appropriée pour la manipulation. L'utilisation de capteurs est une idée que nous jugeons pertinente pour augmenter les capacités d'interaction. L'exemple cité ci-dessus montre qu'il y un décalage important entre l'action réelle, approcher sa main, et la conséquence qui est l'émission d'un son. Donc, avec les capteurs, la *représentation du comportement* n'est plus appliquée et nous ne sommes plus dans la cas d'une interface tangible.

1.2.3 TUI avec des interacteurs passifs

Dans cette partie nous décrivons deux applications utilisant des interacteurs passifs. Pour récupérer les informations de ces interacteurs, plusieurs méthodes existent, parmi celles-ci la capture vidéo et la capture de mouvement. Nous allons décrire deux TUI basées sur ces techniques.

1.2.3.1 Capture d'informations par caméra

L'idée sous-jacente de l'application "Luminous Room" ([UUI99]) est de développer une technique permettant l'échange de données sur les objets plans d'une salle, tels le mur, le sol ou une table. Le plan où l'utilisateur manipule les interacteurs sert d'écran d'affichage du résultat de l'interaction. Underkoffler, Ullmer et Ishii présentent plusieurs domaines d'applications : l'analyse des fluides, le travail collaboratif et l'optique géométrique décrite par la suite.
Pour l'application sur l'optique géométrique(cf fig. 1.14), l'utilisateur dispose d'interacteurs symbolisant des miroirs et des prismes permettant de dévier la lumière. D'autres types d'interacteurs simulent des sources lumineuses. Ce sont des parallélépipèdes disposant de petits symboles sur le dessus. Une caméra acquiert l'image de la scène. D'après l'image, ils détectent les symboles sur les interacteurs. La forme, la couleur et l'association des symboles renseignent sur l'orientation, la position et le type de l'interacteur. Grâce à ces informations, l'ordinateur calcule le rayon lumineux en fonction de la source et des obstacles. Le cheminement du rayon et les mesures (distance et angle) sont projetés sur le support où sont situés les interacteurs (cf fig. 1.14).

Dans cette application, les interacteurs n'ont aucune liaison avec l'ordinateur car les informations sont captées par la caméra. Le système de reconnaissance implique le placement de marqueurs sur les interacteurs. Cette solution a l'avantage de n'imposer aucune contrainte forte sur la forme de l'interacteur. Le résultat de l'action est projeté dans l'espace d'interaction sur le principe de la réalité augmentée. L'utilisation est très intuitive et l'ordinateur est totalement absent aux

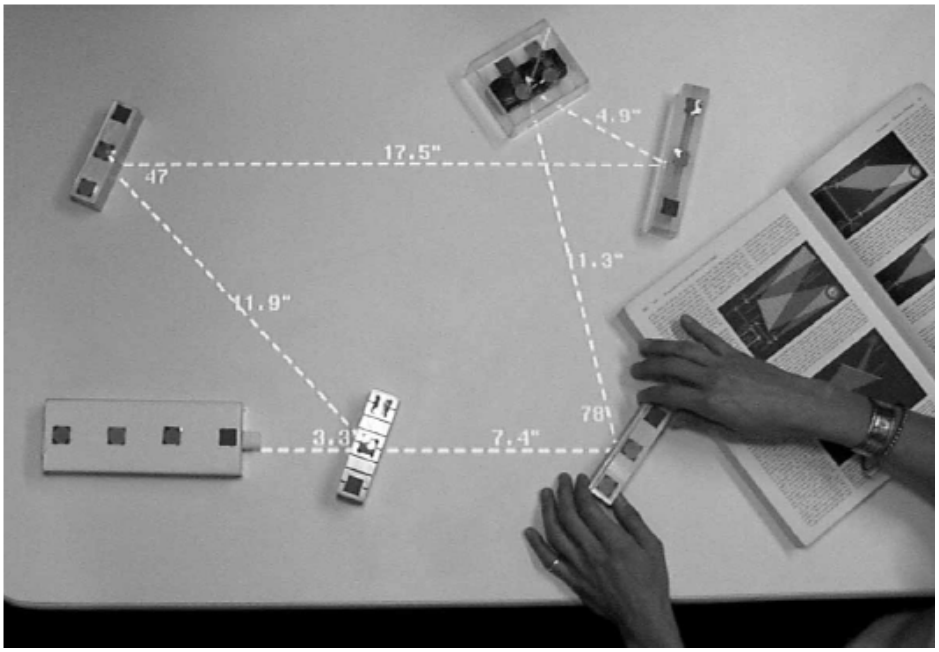


FIG. 1.14 – Photographie de l'application Luminous Room (extraite de [UUI99]).

yeux de l'utilisateur.

1.2.3.2 Capture d'informations par détection de mouvement

Ernst, Schäfer et Bruns créent ([SHEB00], [ESB99]) une interface tangible pour simuler, entre autres, les flux sur une chaîne d'assemblage (cf fig. 1.15). Cette interface est réalisée dans le cadre du projet européen BREVIE (Bridging REality and VIrtuality with a graspable interface). L'interaction est générée par le déplacement de la main, sachant qu'au départ toutes les positions des objets sont connues.

La boîte et les interacteurs ont des positions prédéfinies. Les capteurs du gant (cf fig. 1.16) donnent la position et les actions de saisie de la main. Les informations fournies par les capteurs du gant et la connaissance des positions des interacteurs permettent de calculer les déplacements des objets.

Les interacteurs nommés "Twin-objects" (objets jumeaux) par les auteurs sont des blocs, des tapis et des chaînes. Dans cette application tous les interacteurs n'ont pas la même fonction. Par exemple, on utilise un type différent d'interacteur pour indiquer le fonctionnement d'une chaîne ou la chaîne elle-même. Il y a ici deux types d'interacteurs : les interacteurs de type tâche et ceux de type objet. L'interacteur de type tâche permet de symboliser une action (la direction). L'in-

FIG. 1.15 – L'interface du projet BREVIE (extrait de [Bru98]).

teracteur de type objet fait référence à un objet (la chaîne). Ce principe augmente les possibilités d'actions en utilisant le principe des interfaces tangibles.

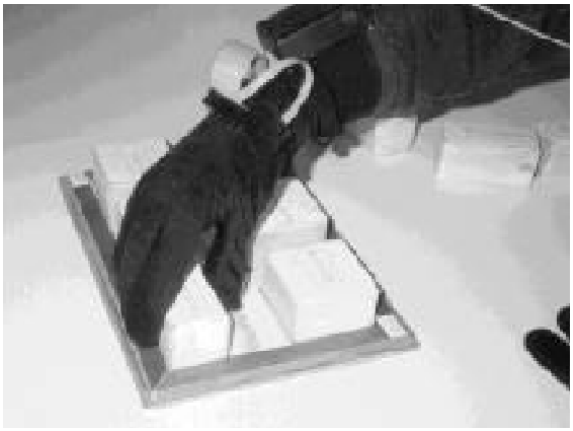


FIG. 1.16 – Photographie du gant du projet BREVIE (extrait de [SHEB00]).

Cette application permet d'utiliser une large variété d'interacteurs avec la précision et la facilité d'utilisation qu'offrent les capteurs. Leur nombre est réduit car l'application capture seulement les mouvements de la main. En revanche, l'utilisateur doit toujours débuter l'application en mettant tous les interacteurs dans une position connue. Á l'instar du projet précédent, la Luminous Room, la forme des interacteurs n'a pas été contrainte par le système de capture contrairement au système composé d'électronique. Les interacteurs sont de formes variées et proches des modèles virtuels.

1.2.4 Synthèse

Les interacteurs sont une représentation réelle (tangible) de la donnée, mais peuvent également représenter une action (cf. 1.2.3.2). Dans le processus de créa-

tion d'une interface tangible, nous préconisons donc de prendre en compte la tâche et l'utilisateur. Nous devons étudier la tâche pour la comprendre et connaître toutes les actions possibles. Nous devons identifier l'utilisateur pour connaître les codes qui sont liés à son domaine. Après avoir défini la forme générale de l'interacteur, nous devons créer le système permettant de connaître leurs états (positions, couleurs, assemblage, et caetera).
Nous avons décrit deux types de systèmes : les interacteurs passifs et actifs. L'utilisation d'interacteurs actifs permet un traitement plus simple et plus rapide des informations. En revanche, la forme de l'interacteur est moins libre à cause de la contrainte liée aux composants électroniques et les solutions à base de capteurs sont onéreuses. L'utilisation d'interacteurs passifs diminue le coût de fabrication et réduit les contraintes sur la création de la forme. Par contre, le système de capture est souvent complexe à réaliser et requiert des traitements informatiques plus coûteux en temps de calcul.

Chapitre 2

Manipulation géométrique de modèles 3D

Sommaire

Au cours de ce chapitre, nous étudions tous les types d'Interfaces d'Interaction Homme-Données (I2HD) conçues pour la manipulation géométrique que nous avons identifiés. Cette étude a permis de proposer une taxinomie basée sur les critères nous semblant les plus importants pour la réalisation d'une I2HD centrée sur la manipulation géométrique de modèles 3D. Dans la première partie, nous expliquons nos motivations pour la réalisation de cette taxinomie. En seconde

partie nous déterminons ces critères. Dans la troisième partie, nous analysons les applications les plus pertinentes au regard de ces critères. Nous énoncerons en conclusion des recommandations à la conception d'une interface tangible dédiée à la manipulation de modèles 3D.

2.1 Cadre de l'étude

De plus en plus d'applications requièrent une interaction plus directe et plus intuitive avec la donnée numérique. Citons, les travaux [GMK99] de Gorman, Meier et Krummel, qui superposent à une image réelle des informations provenant d'une source numérique en utilisant les principes de la réalité augmentée pour des applications en médecine. Les travaux [BGF00] de Buxton, Fitzmaurice, Balakrishnan et Kurtenbach qui s'appuient sur une visualisation de type grand écran et l'utilisation de capteurs (détections des mouvements des mains et de leurs positions) pour interagir plus naturellement pour la conception automobile. Ou encore les travaux [LMB+02] de Lécuyer, Megard, Burkhardt, Lim, Coquillart, Coiffet et Graux qui évaluent les différentes rétroactions (feedbacks) dans le processus d'interaction.

Les facteurs d'une I2HD à étudier pour tendre vers une interaction plus naturelle et plus performante sont les composantes d'action, les composantes de perception, la tâche et l'utilisateur. L'objectif de ce chapitre est de décrire différentes solutions utilisées en IHM pour permettre la manipulation géométrique. Cette étude sort du cadre des I2HD de type Interface Tangible, car nous pensons judicieux de nous inspirer des qualités d'autres I2HD pour les adapter aux interfaces tangibles.

Pour étudier les I2HD existantes, le site web [Bux02] de Buxton est riche et pertinent. Il recense un grand nombre de périphériques matériels et est régulièrement mis à jour. Plus spécialisés, Subramanian et Ijsselsteijn dans [SI00] décrivent et critiquent les applications permettant d'agir sur les six degrés de liberté et d'exécuter des actions atomiques. Pour classer les I2HD, Greenberg propose une taxinomie propre au domaine IHM dans [Gre97] basée sur l'utilisation, le contexte, les caractéristiques humaines, l'interface et le système.
Toutes ces études sont riches pour l'évaluation d'une I2HD, mais dans notre cas elles ne décrivent pas assez précisément les actions possibles avec le périphérique matériel. Or, dans les interfaces tangibles, un point crucial dans la conception est la partie tangible. La plus grande partie de l'interaction est concentrée dans l'interacteur : la cohérence, les actions disponibles, l'utilisation des schèmes[1] . C'est pourquoi nous avons voulu étudier d'une manière plus précise l'influence du périphérique matériel au sein des I2HD. À la manière de Subramanian et Ijsselsteijn [SI00] nous recensons les applications nous semblant les plus pertinentes pour faire ressortir les topologies de techniques logicielles et matérielles mises en

[1] Schèmes n.m. :Représentation psychologique simplifiée, intermédiaire entre l'image concrète et le concept abstrait.

œuvre pour permettre l'interaction.

2.1.1 Finalité de l'étude

Cette étude a deux objectifs. Le premier est de décrire des techniques d'interaction permettant la manipulation géométrique. Les techniques d'interactions sont les possibilités d'actions proposées à l'utilisateur avec les périphériques matériels et leurs composantes logicielles. Le deuxième objectif est d'évaluer la pertinence des choix matériels et logiciels des applications d'un point de vue utilisateur. Dans ce but, après avoir présenté notre vision de la manipulation de données, nous définissons certains termes nécessaire à cette étude. Ensuite, nous discutons des choix matériels et logiciels de certaines applications.
Toutes ces démarches sont faites en gardant à l'esprit que nous nous plaçons dans un contexte humano-centrique.
Nous avons retenu les applications que nous jugeons pertinentes et représentatives des différentes manières de concevoir et d'aborder l'interaction. Notre but n'est pas de répertorier toutes les applications, mais de décrire celles que nous jugeons les plus profitables à nos travaux.

2.1.2 Description de l'étude

Nous analysons les facultés et les moyens de manipuler des modèles géométriques 3D en fonction des capacités du périphérique. Le repère utilisé pour les descriptions est le repère orthonormé $(\mathrm{O}, \overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})$ où $(\mathrm{O}, \overrightarrow{x}, \overrightarrow{y})$ est le plan parallèle à l'écran et $(O, \overrightarrow{x}, \overrightarrow{z})$ le plan parallèle à la table.

Notation 1 *Nous notons t_x, t_y et t_z les trois translations suivant les axes respectifs $\overrightarrow{ox}, \overrightarrow{oy}, \overrightarrow{oz}$ et r_x, r_y et r_z les trois rotations autour des axes respectifs ox, oy, oz.*

Cette thèse utilisant des termes mécanique et informatique, nous sommes obligés de définir les termes de degrés de liberté, mobilité et mobilité interprétée pour être en accord avec les deux domaines. En effet, en informatique il n'est pas choquant de parler d'un objet possédant un nombre de degré de liberté supérieur à six, or ceci n'est pas concevable en mécanique.
Dans cette thèse, lorsqu'un objet dans l'espace est mobile suivant au moins un de ses six degrés de liberté, nous parlons de **degré de liberté libéré** et de **mobilité**.

Définition 1 *Un **degré de liberté libéré** par rapport à un objet correspond à un mouvement possible suivant un axe donné.*

Par exemple, la souris possède trois degrés de liberté libérés : les deux translations sur le plan $(\mathrm{O}, \overrightarrow{x}, \overrightarrow{z})$de la table et la rotation r_z.

Définition 2 *Le terme de* ***mobilité*** *indique la possibilité d'interagir suivant un degré de liberté sur le périphérique. Dans le cas d'un périphérique composé de plusieurs éléments, son nombre de mobilité est la somme des degrés de liberté libérés de chaque sous-partie.*

Définition 3 *Le terme* ***mobilité interprétée*** *indique les mobilités qui sont transmises à l'application et qui sont interprétées.*

Par exemple, la souris est composée d'une ossature (la coque) et d'une roulette. La souris dispose de quatre mobilités : la roulette a un degré de liberté libéré et l'ossature (sur la table) a trois degrés de liberté libérés. Nous avons seulement trois mobilités interprétées informatiquement. En effet la rotation r_y de la souris n'est pas interprétée.

Une I2HD est utilisée dans deux espaces en même temps : l'espace réel et l'espace virtuel. L'utilisateur agit dans l'espace réel en manipulant le périphérique, pour décrire cette action nous parlons de geste réel. Le résultat de cette action est une modification d'un ou de plusieurs paramètres de la tâche (couleurs, taille, poids) ou de l'application (mode vue, mode déplacement).
Par exemple, les paramètres de l'application peuvent être la couleur et la position des objets, le point de vue de la caméra.
Entre le nombre de possibilités du périphérique et le nombre d'actions contenues dans l'I2HD la différence est conséquente pour l'application. Pour permettre à l'utilisateur d'accéder à toutes ces fonctionnalités, les I2HD s'appuient sur les icônes, les menus et ce que nous appelons les **contextes**.

Définition 4 *Le* ***contexte*** *est un état permettant de modifier la traduction par l'application du comportement d'un périphérique. Ce contexte peut être accédé de manière matérielle ou de manière logicielle.*

Par exemple, un contexte matériel peut être un bouton ou une touche du clavier et un contexte logiciel peut être un état comme le mode caméra ou le mode dessin.

Définition 5 *Une* ***action réelle*** *est l 'action exécutée sur le périphérique par l'utilisateur et éventuellement un contexte.*

Définition 6 *Une* ***action virtuelle*** *est la répercussion d'une action réelle sur la donnée virtuelle.*

Pour permettre de décrire le geste réel de l'utilisateur et son résultat dans l'application nous définissons une fonction de transformation.

Par exemple, on considère la souris et l'utilisateur qui exécute une translation de la souris suivant l'axe $\overrightarrow{oz}$ avec l'icône de rotation activé. Alors, la mobilité

interprétée est t_z, le contexte est l'icône de rotation et le résultat est une rotation de la donnée autour de l'axe $\overrightarrow{oz}$.

Nous étudions par la suite des I2HD en identifiant pour chacune les actions réelles et les actions virtuelles associées.

2.2 I2HD pour la manipulation géométrique

Dans cette étude les I2HD permettent la manipulation géométrique. Nous qualifions de manipulation géométrique le déplacement (translations, rotations) d'objets dans le monde virtuel. Les périphériques sont présentés en fonction de leurs mobilités interprétées. Nous cherchons à étudier le lien entre le nombre de degrés de liberté de la composante d'action, les métaphores d'interactions et une interaction simple à apprendre. Il est à noté que pour chaque périphérique matérielle nous avons choisi une application logicielle particulière. Cette association influence fortement les métaphores d'interaction et peut changer le comportement de l'utilisateur dans l'interaction. Nous avons essayer de prendre des articles ou des logiciels où l'interaction nous semblait intéressante.

2.2.1 Souris et Geomview

La souris est un périphérique matériel couramment utilisé. Sa manipulation est devenue transparente dans le mécanisme d'interaction 2D : l'utilisateur pense à son action sur l'interface logicielle et non à la manipulation à réaliser à l'aide de la souris.
Pour notre étude nous devons compléter ce périphérique pour obtenir une I2HD. Nous y associons donc l'interface logicielle Geomview [Geo]. Geomview est un freeware sous licence LGPL. Il permet de manipuler des objets 3D à l'aide de la souris et il dispose d'une interface logicielle simple (cf. photo. 2.1). En effet les actions possibles se limitent à la visualisation, les menus et les icônes sont donc peu nombreux en comparaison à des logiciels comme Maya, Catia où même Word.

Pour les rotations, cette I2HD se base sur la technique de la sphère virtuelle ([CMS88]). Cette technique permet de faire tourner un objet dans l'espace avec seulement deux mobilités interprétées. En revanche elle ne permet pas une manipulation rapide de la rotation autour de l'axe $\overrightarrow{oz}$. La solution proposée dans le logiciel Geomview est d'utiliser le bouton du centre de la souris pour tourner autour de cet axe. Le contexte "bouton du milieu" permet les déplacements suivant l'axe $\overrightarrow{oz}$.

Ici, la souris dispose de deux mobilités interprétées car la souris étudiée ne possède pas de molette. Les actions réelles sont réduites avec seulement deux mobilités. La souris est donc un périphérique possédant peu d'action réelle et elle s'utilise avec interface logicielle proposant quasiment de toutes les actions virtuelles.

l'I2HD doit donc avoir un nombre de contextes importants (interface logicielle riche) pour permettre la génération de tous les actions virtuelles.

FIG. 2.1 – Photographie d'écran du logiciel Geomview.

L'utilisation de plusieurs contextes augmente la durée d'apprentissage : il faut les mémoriser, les connaître et les comprendre. Son utilisation est simple pour interagir avec une interface à deux dimensions, car les déplacements de la souris sont proches des déplacements du curseur (objet virtuel associé). C'est pour cette raison, que la souris reste le périphérique le plus utilisé pour interagir avec une interface logicielle (graphique) 2D. Toutes les métaphores d'interaction et les connaissances sont dans l'interface logicielle. Donc, la partie la plus importante dans la composante d'action est l'interface logicielle, en effet on peut très facilement remplacer la souris par un stylo optique sans modifier la qualité de l'interaction.

2.2.2 Rockin'Mouse

La Rockin'Mouse est un périphérique matériel de forme similaire à une souris (cf. photo. 2.2) et possédant quatre mobilités : deux translations dans le plan du bureau et deux inclinaisons avant/arrière et droite/gauche. Les inclinaisons peuvent être assimilées à des rotations d'environ 100 degrés.

Pour cette I2HD le nombre de contextes ne peut être comparé aux autres I2HD. En effet, dans l'application présentée dans l'article [BBKf97], l'utilisateur ne peut exécuter que des translations, dans l'espace, de l'objet. Comme il y a moins d'actions virtuelles, il y a forcément moins d'actions réelles. Le nombre de contextes est donc inférieur à celui des autres applications car l'utilisateur ne peut pas effectuer de rotations des objets. Néanmoins nous avons choisi d'étudier cette I2HD car elle est une évolution de la souris.

Dans [BBKf97], les auteurs comparent la Rockin'Mouse et la souris sur les translations d'un objet dans un espace 3D. Lors du test, les sujets, un panel de quatorze personnes, devaient placer une sphère dans une boîte en utilisant

FIG. 2.2 – Photographie de la Rockin'Mouse.

uniquement des translations (t_x, t_y, t_z). La Rockin'Mouse permet d'atteindre l'objectif trente pour-cent plus rapidement qu'avec la souris. Le gain est dû à la manipulation suivant trois degrés de liberté sans changer de contexte. L'utilisateur se concentre uniquement sur la succession de déplacements et la sélection de l'objet avec le bouton gauche.

La manipulation de la Rockin'Mouse est plus rapide et plus intuitive pour les translations car la métaphore n'utilise qu'un seul contexte et les mobilités interprétés sont proches des actions virtuelles. En effet les gestes de translations et d'inclinaisons avec la souris sont facilement associables aux déplacements (translations) de l'objet dans le monde virtuel. L'interaction s'appuie donc sur un comportement naturel pour les translations. Pour proposer des actions supplémentaires, par exemple les rotations, nous pensons que la Rockin'Mouse sera utilisée comme une souris classique, elle servira à actionner les menus et les icônes. L'I2HD aura donc une interface logicielle basée sur des menus et des icônes ayant pour avantage une possibilité d'actions quasi-illimitées et pour inconvénient un nombre de contextes croissants donc plus long à apprendre.

2.2.3 ToolStone

Dans [RS00], les auteurs présentent un périphérique de forme parallélépipédique (cf. photo. 2.3). À l'intérieur se trouvent trois bobines magnétiques qui calculent les angles d'inclinaison et la position. La ToolStone se manipule sur une tablette qui fait le lien entre elle et le PC. Le parallélépipède doit toujours rester en contact avec cette tablette. C'est un périphérique avec cinq mobilités interprétées car il n'y pas d'élévation possible. La ToolStone dispose de boutons à la surface du parallélépipède. Cette application aussi ne permet pas les déplacements suivant les six degrés de liberté, mais s'appuie sur un concept intéressant.

Dans cette I2HD le concept d'objet dominant/dominé est appliqué au périphérique. La Toolstone (objet dominé) s'utilise avec un autre objet (objet dominant) pour augmenter l'interaction globale et permettre de nouvelles actions plus simplement. Par exemple, associé avec un stylo, la ToolStone peut s'utiliser comme une gomme ou peut permettre de sélectionner une couleur.

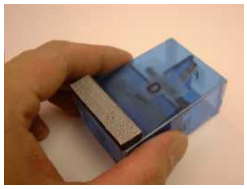


FIG. 2.3 – Photographie d'une ToolStone (extrait de [CSL]).

Pour passer en mode manipulation, l'utilisateur sélectionne l'objet virtuel et maintient la sélection (de la même manière qu'avec une souris). Les gestes exécutés avec la Toolstone sont reportés directement sur l'objet virtuel. Par directement, nous signifions que le résultat est quasi-identique à l'action, par exemple une rotation de la Toolstone pour une rotation de la donnée. Avec les ToolStone les métaphores d'interaction sont simples, elles n'appuient sur un contexte et les actions réelles sont presque identiques aux actions virtuelles. Dans ce cas la manipulation semble directe et donc plus rapide à apprendre.
L'idée intéressante de cette I2HD est la notion d'objet dominant/dominé. La solution technique choisie est rapide et simple à analyser par l'application. En revanche, le contact avec la tablette ne permet que cinq mobilités.

2.2.4 SpaceMouse

Tous les périphériques décrits précédémment et par la suite sont des périphériques isotoniques. Ainsi, les déplacements des composantes d'action sont du même ordre de grandeur que les déplacements de la donnée virtuelle controlée. La SpaceMouse [Spab] (fig. 2.4) utilise un mode de contrôle isométrique dont le fonctionnement est similaire à un capteur d'effort. Dans ce cas, les forces exercées sur la souris sont traduites en déplacement.

La spaceMouse est un périphérique statique disposant de 6 mobilités interprétées (trois rotations et trois translations). Il est compatible avec une centaine de logiciels mais possède tout de même des lacunes. Abolmaesumi, Salcudean, Zhu et Al. [ASZ+01] ont développé une interface pour controler un robot en utilisant la SpaceMouse. Dans leur conclusion, ils reprochent l'absence de retour haptique à la SpaceMouse et pensent améliorer l'interaction en utilisant le Voice-Coil [SP97] qui est un périphérique haptique.
Bloomfield et Deng et Al. [BDW+03] ont comparé des gants CyberGloves, le Phantom (décrit dans la suite) et la SpaceMouse pour les tâches de démontage.

FIG. 2.4 – Photographie de la SpaceMouse (extrait de [Spab]).

Leur conclusion laisse penser que le Phantom est plus performant grâce à son retour haptique et son mode de déplacement isotonique. Le mode isotonique leur semble plus pertinent dans le cas d'une scène réduite où les déplacements sont restreints.
Zhai constate également que les interfaces isotoniques sont d'un apprentissage plus aisés pour les tâches de manipulation et qu'elles souffrent d'inconvénients comme des mouvements restreints dans l'espace et la difficulté d'aquérir les informations venant du périphérique. À l'inverse, les interfaces isométriques sont plus pertinentes pour les espaces de grandes dimensions et sont plus faciles à acquérir.

2.2.5 Cubic Mouse

La Cubic Mouse [FPWa00] (cf. photo. 2.5) est un cube aux faces traversées par trois tiges. Le cube est équipé d'un capteur de position qui permet de connaître son orientation ainsi que ses mouvements, soit l'état de ses six degrés de liberté. La tige et le cube forment une liaison pivot glissant (rotation autour d'un axe et translation suivant ce même axe). Ce périphérique matériel possède douze mobilités interprétées (cf. def. 3, page 28) pour l'interaction.

Les valeurs des degrés de liberté du cube déterminent l'orientation et la position de la scène (monde virtuel). Dans l'application de l'article [FPWa00], les tiges servent à manipuler (déplacement suivant un axe) des plans de coupe orthogonaux et la Cubic Mouse à déplacer l'objet sélectionné.

Ici, l'action rélle est identique à l'action virtuelle à une homothétie près. La manipulation de la scène est une manipulation semi-directe, ce qui signifie que l'application n'utilise pas de contexte et que les actions réelles sont quasiment identiques aux actions virtuelles. Nous pensons que lier les comportements du périphérique et de la donnée numérique conduit à une meilleure interaction. Ceci s'est confirmé lors des présentations de la Cubic Mouse, où les auteurs ([FPWa00])

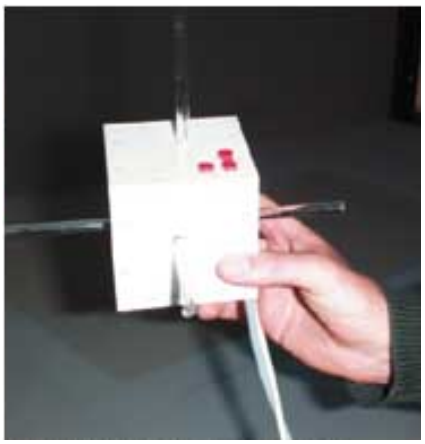


FIG. 2.5 – Photographie de la Cubic Mouse (extraite de [ERC]).

ont constaté que les utilisateurs ne se focalisent pas sur le cube mais sur la donnée affichée, et réalisent efficacement les manipulations.
En conclusion, cette I2HD est intéressante pour la manipulation d'une scène car elle se base sur des gestes intuitifs. La Cubic Mouse et la donnée numérique ont toujours la même orientation. Les sens visuel (stimulé par l'affichage) et kinesthésique (stimulé par la Cubic Mouse) de l'utilisateur sont en concordance. Nous remarquons que les contextes sont absents car il n'y qu'un objet manipulé à la fois et il s'agit de l'objet affiché. Pour manipuler plusieurs objets, l'I2HD devra permettre une sélection d'objet.
Par contre, pour permettre une interaction semi-directe, l'interaction se limite à la manipulation des plans et de l'objet. L'utilisateur ne peut pas manipuler plusieurs objets en même temps et ne peut pas sélectionner des objets avec la Cubic Mouse.

2.2.6 Phantom et FreeForm

Dans le domaine de la réalité mélangée, il y a une plus grande variété dans les dispositifs. La société ReachIn ([RCT]) a créé un système (cf. photo 2.6) permettant de combiner les espaces d'action et de perception. Ce système peut être utilisé dans différentes applications : la simulation et la formation médicale, les maquettes numériques.

Le Phantom de Sensable Technologie ([Che99a],[CC99]) est un périphérique matériel à retour d'effort agissant suivant les six degrés de liberté.

Nous avons utilisé le Phantom dans deux configurations différentes et donc deux métaphores d'interaction différentes.

Le premier dispositif de ReachIn est constitué d'un Phantom positionné sous

FIG. 2.6 – Photographie du Phantom.(extraite de [PHA]), dispositif ReachIn (extrait de [RCT]).

un miroir semi-teinté sur lequel une image est projetée en stéréoscopie (cf. photo 2.6). L'utilisation du miroir semi-transparent crée une interface où les images et le périphérique matériel sont placés dans le même espace. L'utilisateur peut voir et toucher l'objet dans l'espace où il agit. Le scalpel virtuel et le Phantom semblent confondus pour l'utilisateur.
Le deuxième dispositif est le logiciel FreeForm, un écran classique et le phantom disposé à coté. Ce logiciel permet de faire de la sculpture virtuelle. Le Phantom sert d'outil pour enlever ou ajouter de la matière. Les mouvements exécutés sur le Phantom sont reproduits à l'identique sur l'outil virtuel. Nous avons évalué le Phantom avec le logiciel FreeForm. L'ensemble nous a été prêté par le CRT ESTIA-innovation.

La grande différence entre ces deux applications est l'utilisation de la *synchronisation* - en utilisant le miroir - de l'action et du résultat. Dans ce cas, nous trouvons l'interaction facile à appréhender et rapide à comprendre. En revanche, sans la *synchronisation*, avec le logiciel FreeForm, l'interaction nous semble moins directe et un peu plus difficile à appréhender. En effet nous agissons dans un espace (à notre droite) et regardons le résultat de l'action sur un écran (positionné en face de nous).

A travers cette I2HD nous avons pu percevoir le rôle de la *synchronisation* et de la métaphore d'interaction. À partir d'un dispositif matériel identique (le Phantom), l'I2HD utilisant le concept de synchronisation de la perception et de l'action nous a semblé plus performante que celle ne l'utilisant pas. Par contre, le nombre d'actions réalisables avec la synchronisation est réduite car elle se limite au nombre d'actions de l'outils simulé.

2.2.7 Modèle de Segal

Dans [SSWF00], Sutphen, Sharlin, Watson et Frazer modernisent une interface tangible réalisée vingt ans auparavant : le «Segal model». Le but de cette I2HD est de permettre à l'utilisateur de créer un monde virtuel en édifiant des murs et en plaçant des personnages.

FIG. 2.7 – *gauche :* photographie de l'interface dans le monde réel, *droite :* le résultat dans le monde virtuel(extrait de [SSWF00]).

L'interface tangible est une table formée d'une grille de 24 colonnes et 16 lignes. Les murs sont édifiés à partir de panneaux insérés par l'utilisateur dans la table (cf. photo. 2.7). En outre, le monde virtuel peut être peuplé avec des entités et des personnages virtuels par des moyens physiques (interacteurs) ou avec un éditeur logiciel. Le modèle physique est transcrit et transféré automatiquement en un modèle virtuel 3D. Ce modèle est affiché avec le moteur graphique 3D d'«Half Life». Les auteurs concluent que ce type d'interface tangible est plus simple à utiliser et à appréhender que les I2HD basées sur l'utilisation de la souris. La manipulation dans le monde réel est identique à la manipulation dans le monde virtuel.

La manipulation est réduite à l'utilisation des panneaux, il n'y a aucun contexte. L'utilisateur peut déplacer les panneaux sur le plateau et leur donner une orientation en les insérant.

Dans les I2HD classiques, l'action virtuelle prime sur l'action réelle, l'utilisateur réalise un geste avec la composante d'action dans le but de réaliser une tâche virtuelle. Ici, l'action réelle est plus importante que l'action virtuelle qui n'intervient qu'une fois le montage terminé. L'utilisateur réalise son action sans réfléchir au résultat dans le monde virtuel. Il pense et agit par rapport aux données réelles. Nous pouvons presque considérer que la métaphore d'interaction est quasi-inexistante pendant la manipulation. Elle n'existe qu'une fois le montage terminé : elle transcrit le montage réel en un monde virtuel (cf. photo. 2.7).

2.3 Synthèse, taxinomie

Lors de l'étude des I2HD précédentes, nous avons identifié trois principales manières d'interagir que nous nommons : l'interaction symbolique, l'interaction semi-directe, l'interaction directe. L'interaction symbolique s'appuie essentiellement sur l'utilisation de métaphores logicielles (icône et menu) alors que l'interaction semi-directe et directe s'appuient sur des métaphores matérielles.

2.3.1 Interaction symbolique

Nous introduisons la notion d'interaction symbolique pour décrire l'interaction reposant sur l'utilisation de symboles (icônes) et de menus. L'écart entre l'action réelle réalisée sur le périphérique et sa répercussion sur l'objet dans le monde virtuel est important.

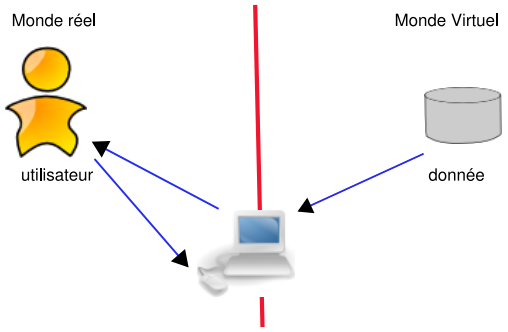


Fig. 2.8 – Illustration de l'interaction symbolique.

Dans l'interaction symbolique (cf. fig. 2.8), l'utilisateur doit concevoir ses actions en utilisant principalement l'interface logicielle au travers d'icônes et de menus ; l'interface matérielle ne servant qu'à opérer sur l'interface logicielle.
Il existe une grande séparation entre l'utilisateur, l'interacteur et l'application. L'I2HD implique un apprentissage relativement long car l'utilisateur doit comprendre et apprendre les symboles. En revanche, l'avantage de ce type d'interaction est de permettre une possibilité d'actions quasi infinie. En effet, les actions ne sont limitées que par l'interface du logiciel.
Le développement logiciel est simplifié, car la souris (principal périphérique de cette catégorie) est maîtrisée par les utilisateurs et les développeurs. Avec, l'ajout du clavier et des menus contextuels, il est facile pour le développeur de proposer les actions que l'application requiert.

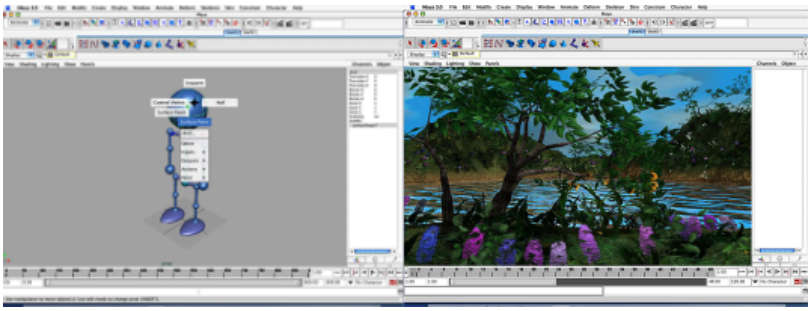


FIG. 2.9 – Captures d'écran du logiciel Maya ([Maya]).

Les meilleurs exemples sont les logiciels de CAO (Catia de Dassault System) et de modélisation (Maya d'Alias WaveFront, [Mayb], fig. 2.9) où le nombre d'actions possibles est très élevé.
Les métaphores d'interaction associées à ce type d'interaction disposent d'un grand nombre de contextes. Les actions réelles sont très différentes des actions virtuelles. La souris illustre parfaitement ce type d'interaction.

L'interaction symbolique est basée sur des métaphores d'interaction s'appuyant sur une action (click, déplacement, etc.) et un contexte (exemple : souris) très différent de l'action virtuelle.

2.3.2 Interaction semi-directe

Nous introduisons la notion d'interaction semi-directe pour décrire les I2HD où l'interface matérielle est conçue par rapport aux schèmes de l'utilisateur que l'on souhaite stimuler. Le terme de schème désigne un comportement ne nécessitant pas d'effort cognitif important.
Cette deuxième catégorie introduit une différence notable avec l'interaction symbolique : l'objet de l'interaction.
Dans l'interaction symbolique, l'utilisateur interagit avec l'interface logicielle pour modifier les données numériques. Tout le travail de conception se fait alors sur l'interface logicielle ; l'interface matérielle servant seulement à déclencher les évènements. Dans l'interaction semi-directe, l'interface matérielle devient importante car une partie des actions est contenue dans le périphérique matériel. Elle devient plus présente que l'interface logicielle. Au travers de l'interface matérielle, les actions les plus importantes sont accessibles par des gestes intuitifs sur le périphérique. Les espaces d'action et de perception étant séparés, il y a un contrôle visuel important de la part de l'utilisateur. L'utilisateur doit connaître une partie du processus d'interaction pour exécuter les actions secondaires, les plus importantes faisant appel à des schèmes (souvent la manipulation). L'I2HD permet une concordance (corrélation) entre le geste réalisé et le retour visuel (représenta-

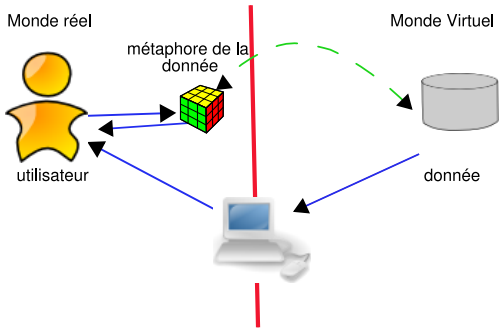


FIG. 2.10 – Illustration de l'interaction semi-directe.

tion visuelle de la donnée). L'utilisateur connaissant les gestes, l'apprentissage est plus rapide et plus naturel. En revanche, disposer les actions de manière directe avec le périphérique matériel limite fortement le nombre d'actions possibles sur la donnée numérique. La conception devient également plus compliquée, car il faut réaliser des périphériques spécifiques pour chaque ensemble de manipulation. Il est toujours possible d'utiliser une interaction symbolique pour les fonctions plus complexes et ainsi garder une possibilité d'évolution pour l'application.

Les métaphores d'interaction associée à ce type d'interaction s'appuient sur un nombre de contextes plus faible voir nul que l'interaction symbolique. Les actions réelles sont proches des actions virtuelles, par exemple, une translation de la composante d'action correspondra à une translation de la donnée. Ce sont donc des interfaces plus rapides à apprendre car le geste réel est représentatif de la *répercussion sur la donnée dans le monde virtuel*. Par exemple, les mouvements de la Cubic Mouse sont reproduits à l'identique sur la donnée.

L'interaction semi-directe est associée à des métaphores d'interaction simple à assimiler. Les actions virtuelles reproduites sur la donnée sont les mêmes actions que celles réalisées avec la composante d'action. Le retour d'informations se situe dans un espace différent de celui de l'action (exemple : Cubic Mouse).

2.3.3 Interaction directe

Nous introduisons le terme d'interaction directe pour décrire les I2HD où l'espace de perception est confondu avec l'espace d'action. Les I2HD de type interaction peuvent utiliser les mêmes périphériques (par exemple : le Phantom) que les I2HD de type interaction semi-directe. En revanche, dans l'interaction directe, l'utilisateur interagit sur la donnée contrairement à l'interface semi-directe où il interagit sur une représentation de la donnée. C'est pour cette raison que nous définissons le terme de "directe" pour ce troisième et dernier type d'interaction. Pour permettre de confondre les deux espaces, les composantes de perception sont

souvent des dispositifs de **R**éalité **M**élangée (Workbench [CFH97]) ou **R**éalité **V**irtuelle (casque virtuel [KTY97]).

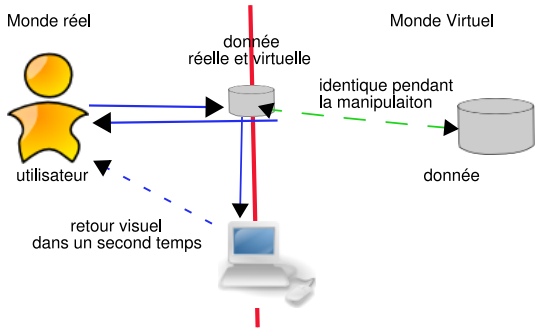


FIG. 2.11 – Illustration de l'interaction directe.

Dans l'interaction directe, l'I2HD utilise les capacités de l'utilisateur au travers de ses schèmes. Elle repose entièrement sur l'interface matérielle. Il s'agit d'une conception humanocentrique. L'interface matérielle a le même comportement que la donnée. L'utilisateur n'a pas la sensation de penser le processus d'interaction, il agit dans le but de réaliser son action et non pas dans le but de faire réaliser son action (par une interface ou un interacteur). Même si les intermédiaires existent (logiciel, gant, projecteur, etc.) ils deviennent transparents pour l'utilisateur. Il manipule l'objet de manière intuitive directement sur la donnée virtuelle, il n'a pas à reporter ses actions sur un objet.
L'interaction directe et semi-directe s'appuient toutes les deux sur des actions réelles proches des actions virtuelles.Par contre, avec l'interaction directe, l'espace d'action et l'espace de perception sont confondus. Par exemple, avec le Phantom, la composante de perception est composée d'un miroir semi-teinté permettant de confondre l'espace de perception et l'espace d'action.

L'interaction directe s'appuie sur des métaphores d'interaction transparentes trés simples à assimiler et l'espace d'action est confondu avec l'espace de perception (exemple : modèle de Segal).

2.3.4 Conclusion

De cette étude, trois types d'interactions ont émergé : l'interaction symbolique, l'interaction semi-directe et l'interaction directe.

- **l'interaction symbolique** : métaphore d'interaction basée sur une action (click, déplacement, etc.) et un contexte (exemple : souris, Phantom, SpaceMouse, ToolStone).
- **l'interaction semi-directe** : métaphore d'interaction reproduisant l'action réalisée avec le périphérique sur la donnée. Le retour d'informations se

situe dans un autre espace (exemple : Cubic Mouse, Phantom et FreeForm, SpaceMouse, ToolStone).
- **l'interaction directe** : métaphore d'interaction transparente, l'espace d'action est confondu avec l'espace de perception (exemple : modèle de Segal, Phantom) et les actions réelles et virtuelles sont quasi-identiques.

Nous pensons que l'interaction directe permet de générer des métaphores d'interaction plus proche de la réalité et donc une utilisation plus rapide à assimiler. En revanche, le nombre d'actions possibles est contraint par les possibilités des périphériques matériels excepté dans le cas d'environnements virtuels utilisant des systèmes immersifs. En effet, dans ces environnements tous les outils sont disponibles, mais non réels. L'interface semi-directe est tout aussi performante et nécessite des moyens matériels moins importants. En effet, cette manière d'interagir ne nécessite pas de dispositif particulier pour confondre les espaces d'action et de perception.
De cette partie, nous avons établi des choix de conception pour la réalisation des interacteurs d'une interface tangible : une interaction directe ou semi-directe pour les actions pertinantes et une interaction symbolique pour les actions secondaires. La composante d'action doit permettre à l'utilisateur d'exécuter de manière directe les actions les plus importantes. En d'autres termes, l'action réalisée sur le périphérique dans le domaine réel a une conséquence identique sur la donnée numérique dans le domaine virtuel. En revanche, rechercher systématiquement une interaction directe serait une erreur dans les cas complexes à réaliser. Par exemple, pour la tâche d'extrusion d'une forme, il est difficile de trouver une représentation physique de cette action.
Pour les actions secondaires, nous préconisons d'utiliser une interaction symbolique plus simple à mettre en œuvre et aux possibilités quasi illimitées.

Pour définir la manière d'interagir (symbolique ou directe ou semi-directe), nous devons décrire et recenser toutes les tâches liées à l'application. Une fois ce recensement réalisé, nous devons analyser la fréquence d'utilisation et la complexité de chaque tâche pour mettre en œuvre une interaction de type directe. La détermination des tâches à accéder de manière directe permettra de définir les propriétés de l'interacteur.

Chapitre 3

SKUA : conception

Sommaire

SKUA, **S**ystème **K**inésique **U**tilisable pour l'**A**ssemblage, est le nom de l'interface tangible que nous avons développée. Le champ d'application choisi est l'assemblage en CAO. Cette I2HD permet de réaliser l'assemblage de pièces CAO. Notre but est de proposer au concepteur[1] un environnement de travail le confrontant à des contraintes d'assemblage qui sont occultées actuellement par les fonctionnalités des logiciels de CAO existants. Nous nous sommes imposés comme contraintes que le système soit d'un coût proche d'un PC standard, soit utilisable sur un bureau et ne nécessitent pas d'équipement (gants, casques).

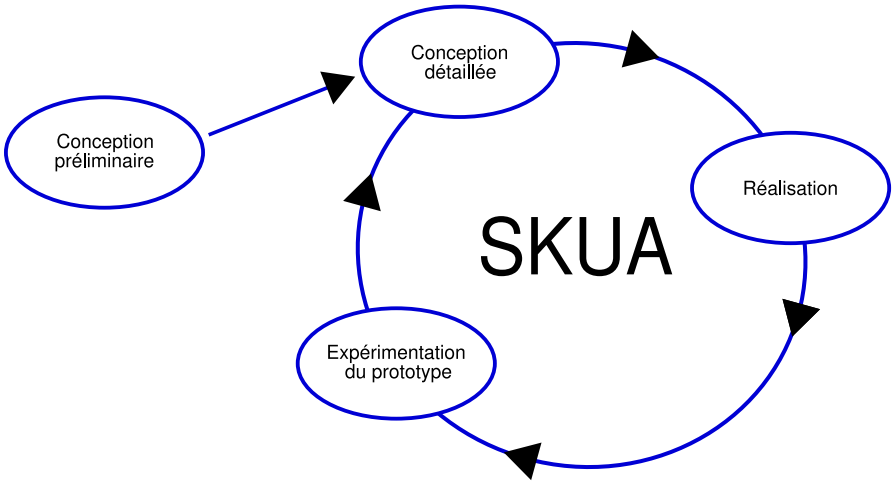


FIG. 3.1 – Schéma du processus de conception de SKUA.

Le processus de conception (cf. fig. 3.1) de SKUA se décompose en quatre phases : la conception préliminaire, la conception détaillée, la réalisation, l'expérimentation. Dans le but de finaliser notre conception préliminaire nous étudions dans la première partie de ce chapitre la tâche de notre I2HD : l'assemblage. Ensuite, en fonction des caractéristiques de l'assemblage nous réalisons la conception détaillée des trois composantes de SKUA : les interacteurs[LGC04],[LGC03],[GLC03] [CCG+04], la communication interacteur-ordinateur et la plate-forme[GC03] matérielle. La plate-forme matérielle est composée : d'un plateau servant de support pour la manipulation et d'un bras telescopique pour positionner le système d'acquisition.

[1] Nous appelons "concepteur" une personne travaillant dans un bureau d'étude (technicien ou ingénieur) ayant pour activité la conception de produits manufacturés sous CAO.

3.1 Définition de la tâche

Dans le second chapitre (cf 1.1), nous avons mis en avant l'influence de la tâche et de l'utilisateur sur la réalisation d'une I2HD.
Les utilisateurs (acteurs, UML) identifiés (cf fig. 3.2, p45) pour l'utilisation de SKUA sont les concepteurs et les monteurs.

Catégories	**Acteurs**
Principal	Concepteur
Secondaire	Monteur

FIG. 3.2 – Tableau des acteurs de SKUA.

Le schéma 3.3 décrit les six tâches principales que doit permettre notre I2HD :
- sélectionner ;
- pointer ;
- déplacer ;
- assembler ;
- supprimer ;
- ajouter.

La fonction «sélectionner» permet d'associer une pièce CAO et un interacteur.
La fonction «pointer» permet de sélectionner les actions dans l'interface logicielle.
La fonction «déplacer» permet de déplacer dans l'espace virtuel la pièce CAO.
La fonction «assembler» permet d'assembler mécaniquement deux pièces CAO.
La fonction «supprimer» retire une pièce CAO de la scène.
La fonction «ajouter» permet d'ajouter une pièce CAO dans la scène.

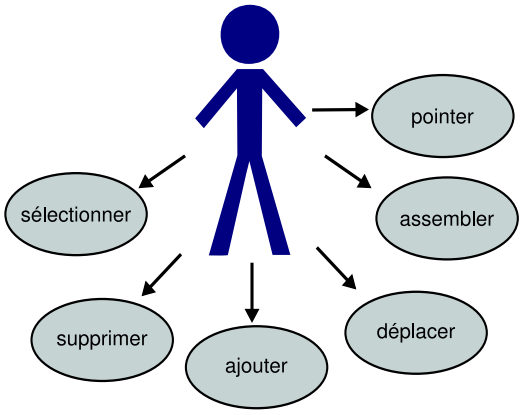


FIG. 3.3 – Les cas d'utilisation de SKUA.

Des quatre tâches précédemment citées, la plus utilisée par l'utilisateur est «assembler». La conclusion de notre taxinomie, nous conduit à chercher une in-

teraction directe ou semi-directe pour cette tâche. L'étude de la tâche n'a pas besoin d'être séparée de l'étude de l'utilisateur puisque l'utilisateur a acquis toutes les connaissances propres à l'assemblage. Nous décrivons l'assemblage de façon à déterminer les caractéristiques des interacteurs à concevoir pour SKUA : leur forme physique et leur comportement.

3.1.1 L'assemblage en CAO

Dans les domaines de la mécanique, du bâtiment, l'électronique, de l'automobile ou de l'aéronautique, l'utilisateur d'un logiciel de CAO a toujours le même objectif : représenter et étudier le fonctionnement d'un objet sans l'avoir réellement fabriqué. Pour représenter un objet, l'utilisateur modélise les pièces puis les assemble. L'assemblage est l'action de réunir des pièces faites pour s'adapter l'une à l'autre, de façon à en composer un tout. Il s'agit ici d'assemblages "numériques" car les pièces n'existent que sous la forme de modèle numérique.
Dans les prochains paragraphes nous soulevons les problèmes liés à l'assemblage en CAO puis nous décrivons une méthode existante rationalisant cette phase.

3.1.1.1 La perception de l'assemblage

Des études sur des cas industriels comme celle de Munroe [Mun95] ont montré que le concepteur dans son environnement (bureau d'étude) n'est pas toujours en mesure d'appréhender son travail. Nous appuyons ce propos en constatant que les environnements proposés par les logiciels traditionnels de CAO ne permettent pas vraiment une véritable immersion du concepteur dans le contexte d'une opération d'assemblage. En effet, le modèle CAO géométrique d'un produit, composé de plusieurs pièces peut être manipulé et assemblé par l'utilisation de la souris et des différentes fonctionnalités du logiciel sans contraintes physiques d'assemblage. Ainsi, l'assemblage sous CAO est relativement « simple » dans sa mise en œuvre pour les opérations traditionnelles permettant de réaliser des rotations, des translations, des mises en positions par coaxialité, par axes et plans de référence. Ces fonctionnalités proposées par les logiciels ne tiennent pas vraiment compte de la difficulté de la mise en œuvre réelle, sur la chaîne de production, des opérations de montage associées à chaque pièce. Les contraintes réelles sont occultées par les fonctionnalités des logiciels de CAO existants. Les fonctions de coaxialité masquent la complexité de la mise en position relative de deux pièces dans le domaine réel. Les problèmes de préhension, d'inaccessibilité ou de collisions des pièces ne sont pas perceptibles par l'utilisateur lors de l'assemblage en CAO. Pourtant, toutes ces difficultés existent réellement lors d'un assemblage.

Étape importante de la fabrication, l'activité d'assemblage représente 30% à 40% du coût de fabrication d'un produit manufacturé et environ 30% des investissements en moyens de production pour les entreprises ([Sch91]). Elle recouvre l'ensemble des problèmes liés à la réalisation des contacts entre les composants

constitutifs d'un produit. Elle est difficilement formalisable de part la diversité et la complexité des opérations de montage et du caractère tacite des connaissances mobilisées.

Au début des années 1980, rassemblés sous le terme générique de "Design for X", des méthodes et des outils sont apparus afin de prendre en compte les différents aspects du produit (fabrication, recyclage, maintenance, qualité) dès la phase de conception. Parmi ces travaux, les méthodes appelées "Design for Assembly" ou "DFA" ([BD83]) ont vu le jour afin d'aider les concepteurs dans leur analyse du produit en cours de développement à l'aide de critères d'assemblage. Ces méthodes font suite à plusieurs tentatives de formalisation dans différents guides d'assemblage, des contraintes et des règles concernant cette activité.

3.1.1.2 Les méthodes Design For Assembly (DFA)

Dans ce paragraphe, nous expliquons, globalement le principe des méthodes "Design For Assembly". Pour une présentation plus complète de ces méthodes, le lecteur pourra se référer à aux travaux de Legardeur [Leg98] et de Redford et Chal [RC94].

Le principal but de ces méthodes est d'identifier des critères d'assemblage et de permettre leur anticipation dès la phase de conception afin d'engager des actions préventives au plus tôt. Ces méthodes sont souvent proposées sous forme de logiciels. Elles permettent de donner un coefficient d'aptitude à l'assemblage manuel d'un produit en se basant sur l'évaluation d'opérations de base concernant le montage : la préhension, la manipulation et l'insertion de la pièce. Deux facteurs de base sont à considérer lors de ces opérations : la quantité d'efforts requis pour l'assemblage et l'efficacité de ces efforts mis en jeu. Pour permettre cette évaluation, l'utilisateur doit fournir en sept étapes des données concernant le produit en cours de conception :

- 1^{ere} étape : dresser la nomenclature (pièces, sous assemblage, etc.) du produit.

- 2^{eme} étape : décrire l(es) opération(s) d'assemblage en désignant le type de fixation associé à chaque pièce (clipsage, soudage, vissage...) et les opérations de montage particulières (graissage, nettoyage, etc.)

- 3^{eme} étape : contrôler la nécessité de chaque pièce. Cette étape permet de vérifier que chaque pièce est indispensable pour le produit en interrogeant le concepteur sur les raisons d'utiliser une pièce nouvelle et favorise une réflexion de ce dernier sur une suppression éventuelle. Plusieurs principes sont alors présentés pour réduire notamment le nombre d'éléments de fixation.

- 4^{eme} étape : décrire la forme générale. Cette étape permet de définir l'enve-

loppe générale de la pièce parmi six formes de base : disque, cylindre court, cylindre long, plaque, bloc, poutre.

- 5^{eme} étape : définir les axes de symétrie de chaque pièce.

- 6^{eme} étape : définir le niveau de difficulté de préhension et de manipulation et le niveau de difficulté d'insertion des pièces. L'utilisateur est amené à faire une critique sur les difficultés liées aux opérations de préhension et de manipulation de chaque pièce. Pour cela, il peut attribuer des caractéristiques à chaque pièce parmi les propositions disponibles (pièces entremêlées, flexibles, collantes, fragiles, glissantes, coupantes ou dangereuses). Un travail identique est nécessaire pour estimer la difficulté d'insertion. Les possibilités de réponses sont par exemple la visibilité réduite, l'accès difficile, le positionnement délicat, la résistance à l'insertion ou le maintien en position.

- 7^{eme} étape : donner les distances entre l'opérateur et les pièces ou outils lors du montage. Cette dernière étape permet de spécifier la distance qu'il y a entre le monteur et le stock de pièces (ou un outil) sur le lieu du montage.

Nous constatons que la méthode DFA permet d'anticiper la phase d'assemblage en proposant un certain nombre de critères d'évaluation pour quantifier des difficultés. Elles sont parfois subjectives (insertion d'une pièce, maintien d'une pièce, mise en position, etc.) et par conséquent occultées par le concepteur. La méthode DFA permet d'identifier des contraintes réelles et physiques liées aux opérations même d'assemblage dès la phase de conception. Nous proposons d'instrumenter une partie des critères et concepts proposés par ces méthodes à travers la création d'une interface tangible basée sur la manipulation d'interacteurs.

3.2 La partie tangible : les interacteurs

Lors de la description des interfaces tangibles (cf. 1.2, p. 15), l'interacteur est présenté comme un des éléments les plus influents de ce type d'I2HD. Après avoir défini les tâches importantes en assemblage nous devons concevoir un interacteur capable de permettre une interaction directe (cf. 2.3.3, p. 39) ou semi-directe (cf. 2.3.2, p. 38) pour ces tâches.

3.2.1 Un interacteur adapté à l'assemblage en CAO

Il existe des interfaces tangibles pour différentes applications et domaines (cf. 1.2). Les exemples les plus proches de nos travaux sont les « Active Cube », le modèle de Segal et le système développé au Laboratoire MERL. Ces applications permettent d'assembler des interacteurs face à face par connections électroniques et excepté l'Active Cube, elles ne fonctionnent pas en temps réel. L'assemblage entre interacteurs par connecteurs ne nous semble pas judicieux. L'utilisateur ne

peut assembler deux interacteurs que face à face, il n'y a ni mise en position ni système de fixation. Dans ce cas aucune notion introduite par les méthodes DFA n'est appliquée. Toutes ces interfaces tangibles ne sont pas conçues et orientées pour la manipulation et l'assemblage.

3.2.1.1 L'étude du comportement des interacteurs

En utilisant SKUA, le concepteur réalise son assemblage en associant un interacteur à une ou plusieurs pièces CAO. Il réalise le montage des pièces CAO en assemblant les interacteurs dans le domaine réel.
En s'appuyant entre autres sur l'étude précédente des méthodes DFA, nous définissons le comportement de nos interacteurs pour l'assemblage.

Nous pensons important de contrôler chaque pièce CAO avec un interacteur afin que leur comportement respectif soit identique. Représenter physiquement tous les types de fixation des pièces CAO sur les interacteurs nous semble difficile. Le nombre de possibilités (par exemple : clipsage, soudage, vissage) augmenterait considérablement le nombre d'interacteurs.
D'après les méthodes DFA, la préhension et la manipulation sont des actions prépondérantes lors de l'assemblage. Par exemple, les difficultés de mise en position relative de deux pièces avant fixation ou encore les difficultés d'insertion d'une pièce par rapport aux autres tels que l'inaccessibilité ou les collisions seront potentiellement identifiables par le concepteur lors de ses manipulations. Nous confrontons l'utilisateur aux contraintes réelles des opérations d'assemblage c'est à dire aux difficultés de mise en position relative des pièces et au maintien de manière conjointe de certains éléments.
Pour rapprocher l'utilisateur de SKUA de l'activité réelle du monteur, nous proposons un système de fixation entre les interacteurs qui est représentatif des manipulations d'assemblage existantes. Une première réflexion nous avait conduits à utiliser un système de fixation à base de velcro. Ceci était simple à mettre en œuvre et donnait une grande souplesse dans le choix de la forme des interacteurs. Dans ce cas, le travail du monteur étant absent lors de l'assemblage de deux interacteurs donc nous avons écarté cette idée.

La solution retenue est basée sur l'utilisation de goujons. Ce système oblige le concepteur à simuler l'assemblage en alignant les pièces et en les maintenant pour insérer les goujons. Pour augmenter les possibilités de positionnement, nous avons choisi de percer les interacteurs à plusieurs endroits.Dans le cas d'assemblage de type emboîtement (exemple : pièces de puzzle) , il s'agit d'un rapprochement ne faisant pas apparaître la notion de montage. En revanche, la solution choisie oblige le concepteur à réaliser un véritable travail de monteur.

En résumé, le comportement des interacteurs doit permettre la mise en place relative et l'utilisation d'un système de fixation simulant l'assemblage. La mise en place relative des interacteurs se réalise par des possibilités d'assemblage multiples. Le système de fixation est externe et non inclus dans l'interacteur.

Ainsi, l'utilisateur est exposé aux véritables contraintes d'assemblage.

3.2.1.2 L'étude de la forme des interacteurs

Les rôles de l'interacteur dans notre interface tangible sont de permettre la manipulation et l'assemblage de pièces CAO. Pour simuler l'assemblage des pièces, la solution précédemment exposée est le perçage des interacteurs. Nous devons trouver une forme adéquate pour la manipulation. Cette forme doit également permettre à l'utilisateur de savoir quelle pièce CAO il manipule.
En nous appuyant sur les études de Ware et Rose ([WR99]), de Kiyokawa, Takemura et Yokoya [KTY00] nous déduisons que la différence de forme entre l'interacteur et la pièce CAO influe peu sur la qualité de la manipulation. Par exemple, l'utilisateur n'aura pas de difficulté à faire tourner réellement un cube pour faire tourner un avion virtuel.

L'utilisateur manipule plusieurs interacteurs en même temps et chaque interacteur symbolise une pièce CAO différente. La variété des formes (cf. fig 3.4) nous conduit à trouver un moyen permettant à l'utilisateur d'identifier l'interacteur associé à la pièce.

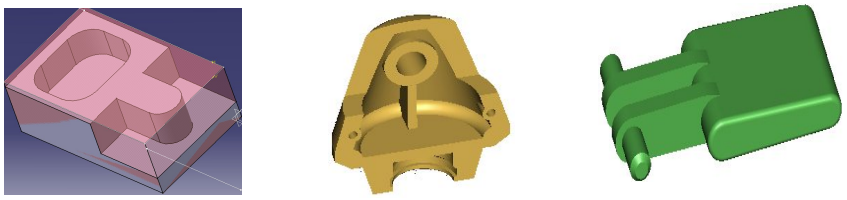


FIG. 3.4 – Illustration de la diversité des pièces de CAO.

Notre réflexion est basée sur les interacteurs car les propriétés de la pièce CAO ne doivent pas être modifiées. Pour concevoir nos interacteurs, nous avions la liberté , entre autre, sur trois caractéristiques : la couleur, la taille et la forme. L'utilisation de codes de couleur (interacteur rouge pour une pièce rouge) fut écartée car nous avons choisi d'utiliser un système vidéo pour repérer nos interacteurs. Nous avons gardé l'utilisation de codes de couleur pour le suivi d'objet.
Lors de l'identification rapide d'un objet nous analysons sa silhouette c'est à dire sa forme générale et son volume. Nous avons donc choisi de travailler sur la forme et la taille.

La silhouette de l'interacteur permet de le classer dans un type de forme simple (par exemple : cube, cylindre). La taille permet de classer les interacteurs d'une même famille entre eux ; ainsi deux objets différents par la taille mais appartenant à la même famille seront associés à des interacteurs de formes identiques mais de

tailles différentes. Le travail intellectuel de l'utilisateur pour identifier l'interacteur associé à la pièce CAO peut être défini comme la charge cognitive. Plus la forme de l'interacteur est éloignée de la forme de la pièce, plus la charge cognitive est importante. En faisant varier la forme et le volume des interacteurs nous pouvons définir une sorte de continuum allant de la maquette de la pièce CAO à une représentation abstraite (par exemple : une sphère). À ce continuum nous pouvons associer deux valeurs : le nombre d'interacteurs et la charge cognitive. La maquette est associée à la charge cognitive la plus faible (voir nulle) et au plus grand nombre d'interacteurs. La représentation abstraite exige la charge cognitive la plus importante et le nombre minimal d'interacteurs.

Entre la représentation exacte et la représentation abstraite, se situe un espace plus vaste que nous définissons par le terme «représentation figurative[2] ».
En utilisant une représentation exacte, c'est à dire une maquette, le nombre d'interacteurs croît avec le nombre de pièces CAO à manipuler. À chaque nouvelle pièce CAO, il faudrait concevoir un nouvel interacteur ce qui serait une aberration.
Dans le cas d'une représentation abstraite basée sur une seule forme d'interacteur (par exemple : sphère ou cube), l'identification des pièces CAO devient trop complexe pour l'utilisateur. Dans ce cas, la compréhension de l'assemblage et de la manipulation des interacteurs est tributaire de la visualisation de la scène virtuelle, or nous voulons que l'assemblage des interacteurs traduise directement le résultat.
Pour conclure, nous pensons qu'il n'est pas impératif d'utiliser une représentation visuelle exacte (forme, volume, couleur). L'utilisation d'une représentation figurative des objets est suffisante et permet de diminuer le nombre d'interacteurs. Lors de l'utilisation des méthodes DFA, nous attribuons des volumes simples aux pièces CAO : sphère, cube ou cylindre. Ces formes suffisent à faire apparaître les contraintes d'assemblage. La sphère, dans les méthodes DFA, sert pour représenter les roulements à bille et pour les liaisons rotules. Hors notre but avec ce jeu d'interacteurs est de représenter principalement les assemblages basés sur les encastrements donc sans mouvement. Ce choix vient du constat que dans les objets courants (souris, tabouret) il y a peu de mobilités. Nous avons donc choisi de créer seulement des interacteurs de formes parallélipipédiques et cylindriques.

3.2.1.3 L'étude de la taille des interacteurs

Après avoir défini une forme et des propriétés de comportement, nous devons définir des tailles. Pour les mêmes raisons que la représentation exacte nous ne pouvons pas utiliser des dimensions proches. Les ordres de grandeur relatifs sont plus adaptés à notre cas que les ordres de grandeur absolus. Au lieu de positionner les tailles sur une échelle absolue, nous les positionnons les unes par rapport aux autres : nous ne dirons pas qu'une taille est petite, grande ou moyenne, mais

[2] figuratif : adj. 1. Qui représente un objet sous sa forme perceptible.

qu'elle est petite par rapport à une autre ou proche d'une autre. Les ordres de grandeurs ne s'expriment alors plus par des relations unaires mais par des relations binaires.
Nous proposons d'utiliser un modèle simple et que nous jugeons suffisant :

- plus petit que, $<$;
- identique à, $==$;
- plus grand que, $>$;
- beaucoup plus grand que, $\gg$;

Cette simplification nous permet de réduire le nombre de tailles d'interacteurs à trois : petit, moyen et grand.

Ce modèle intuitif peut être utilisé par l'utilisateur sans explication préalable. Nous pensons que si nous fournissons un jeu d'interacteurs sur lequel ces règles peuvent être appliquées, l'utilisateur fera naturellement le lien entre la taille des pièces et la taille des interacteurs.

3.2.2 Jeu d'interacteurs proposé

De notre étude, nous dégageons trois contraintes de conception :

- les interacteurs doivent être percés ;
- les interacteurs doivent être de formes parallélépipédiques et cylindriques ;
- les interacteurs doivent être de trois tailles différentes.

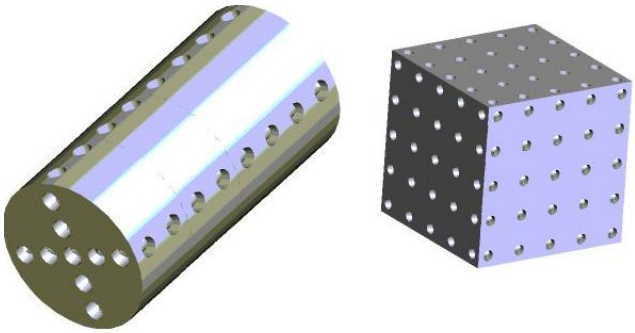


FIG. 3.5 – Exemple d'interacteurs de SKUA.

La conception détaillée a abouti sur un jeu d'interacteurs composé de cylindres et de parallélépipèdes possédant des perçages de 6 mm de diamètre et de 14 mm d'entraxe (cf. fig. 3.5). Les tailles petit, moyen et grand sont respectivement d'un diamètre de 42, 70 et 98 millimètres pour les cylindres et d'arêtes de 42, 70 et 98 millimètres pour les parallélépipèdes. Les différentes profondeurs sont de 14 mm, 28 mm, 42 mm (seulement pour le parallélépipède "petit"), 56 mm, 70 mm (seulement pour le parallélépipède "moyen") et 98 mm suivant la taille. Enfin le matériau doit être léger, solide, non effritable et agréable au toucher.

La figure 3.6,p. 53 regroupe toutes les formes de base des interacteurs de SKUA.

Taille	**Profondeur**	**Cylindre**		**Parallélépipède**	
		Quantité	Identifiant	Quantité	Identifiant
Petit (42 mm)	14 mm	3	$C_{p,1}$	4	$P_{p,1}$
	28 mm	2	$C_{p,2}$	2	$P_{p,2}$
	42 mm	$\emptyset$	$\emptyset$	2	$P_{p,3}$
	56 mm	2	$C_{p,4}$	2	$P_{p,4}$
	98 mm	2	$C_{p,7}$	2	$P_{p,7}$
Moyen (70 mm)	14 mm	3	$C_{m,1}$	4	$P_{m,1}$
	28 mm	2	$C_{m,2}$	2	$P_{m,2}$
	56 mm	2	$C_{m,4}$	2	$P_{m,4}$
	70 mm	$\emptyset$	$\emptyset$	2	$P_{m,5}$
	98 mm	2	$C_{m,7}$	2	$P_{m,7}$
Grand (98 mm)	14 mm	3	$C_{g,1}$	4	$P_{g,1}$
	28 mm	2	$C_{g,2}$	2	$P_{g,2}$
	56 mm	2	$C_{g,4}$	2	$P_{g,4}$
	98 mm	2	$C_{g,7}$	2	$P_{g,7}$

FIG. 3.6 – Tableau des interacteurs de SKUA.

3.3 Méthode de suivi par vidéo

Nous avons décrit précédemment les actions réalisables de manière directe avec les interacteurs (3.3, p. 45) : assembler, ajouter, supprimer et déplacer.
Pour réaliser ces actions, nous devons connaître la position et l'orientation des interacteurs dans la scène réelle. Pour obtenir ces informations, trois techniques existent : les connections électroniques, les centrales inertielles[3] (tracker) et l'utilisation de la capture vidéo. nous les avons décrites dans l'étude concernant les interfaces tangibles (1.2.1.2, p. 16). Lors de la conception des interacteurs, nous avons écarté la solution des connections électroniques car elle imposait trop de contraintes (3.2.1, p. 48), les plus importantes étant le coût et le manque de flexibilité. Les centrales inertielles permettent quant à elles un traitement simple des informations, peu coûteux en ressource machine et précis. En revanche les centrales inertielles tel le kit Xbus de xsense [xse04] sont encore volumineuses (28mm x 40mm x 50mm) et coûteuses (10000$ pour cinq capteurs et le logiciel de développement et 1200$ par capteur supplémentaire). Compte tenu des objectifs du projet ESKUA , la plate-forme expérimentale doit être économique et non intrusive ; nous nous sommes donc orientés sur un système de suivi vidéo en utilisant une webcam en raison de son faible coût. Il existe de nombreuses techniques de suivi d'objet en temps réel. Boykov et Huttenlocher utilisent le filtre de Kalman avec des réseaux bayesiens [BH00]. New, Hasanbelliu et Aguilarles utilisent les détections de couleurs pour extraire des régions [NHMA03]. Smith, Drummond et Cipollales utilisent la détection d'arêtes [SDC04]. Brown, Drummond et Cipolla s'appuient sur un mélange d'approche par modèle 3D et sur la détection des arètes et du filtre de Kalman [BDC00].
Pour le suivi de nos interacteurs, nous avons étudié deux techniques de suivi que nous décrivons par la suite. La première technique, actuellement utilisée dans SKUA, est classique. Elle repose sur l'utilisation de marqueurs. La deuxième s'appuie sur des travaux plus récents et utilise l'approche basée sur les modèles. Nous présentons nos travaux pour mettre en place cet algorithme pour l'instant non abouti mais duquel nous pouvons tout de même tirer des conclusions sur l'utilisation de cette technique pour la capture des parties tangibles de notre TUI. Nous terminons par une discussion sur des travaux futurs qui nous semblent interessants.

3.3.1 Approche basée sur des marqueurs

Suivre un marqueur dont nous connaissons la couleur dans une image est une démarche classique de la capture de mouvement. Cette technique permet d'obtenir des points caractéristiques de manière rapide avec un simple filtre de couleur.

[3] Centrale Inertielle : dispositif muni de gyroscopes, d'accéléromètres et d'un calculateur qui permet à un de connaître la position et la vitesse dans l'espace d'un objet.

3.3.1.1 Description

Actuellement les interacteurs sont assemblés par les cotés. Nous pouvons donc imposer à l'utilisateur de manipuler les interacteurs en laissant toujours la face du dessus visible par la caméra.
Dans la configuration classique d'utilisation de SKUA, il existe deux types d'éléments visibles : les interacteurs et les mains de l'utilisateur. Dans le champ de la caméra apparaissent en permanence les couleurs variant du jaune au rouge (la main) et tous les niveaux de gris du blanc au noir (interacteur et plateau).

Nous avons donc disposé des marqueurs carrés sur le dessus des interacteurs. Ces marqueurs sont de couleur verte pour être extraits plus facilement à l'aide d'un filtre et éviter la confusion avec la main et les interacteurs. Nous avons disposé trois marqueurs sur trois des quatre angles de la face du dessus (cf. fig. 3.8, p. 57). L'utilisation de trois marqueurs, et non de un ou de deux, permet de lever l'ambiguïté sur l'angle entre l'interacteur et l'axe des abscisses.

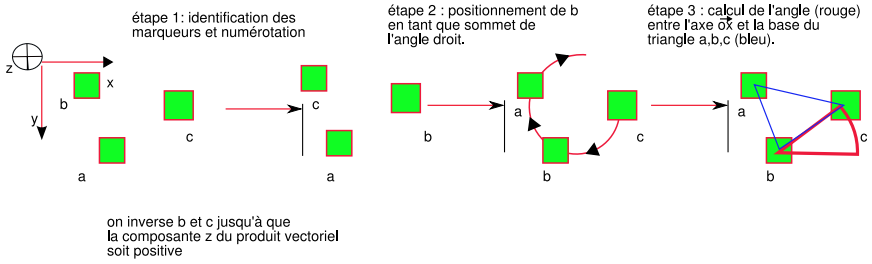


FIG. 3.7 – Description de la numérotation des sommets.

La détection de la position des marqueurs se fait en trois étapes.

Lors de la **première étape**, nous cherchons les marqueurs dans l'image. Nous conservons d'abord tous les éléments de couleur verts pour ensuite reconstituer les régions connexes vertes dans l'image. Pour cela, nous parcourons l'image ligne par ligne jusqu'à rencontrer un pixel vert qui sera le point de départ de la région. Nous réalisons par la suite une expansion de régions circulaires et uniformes en quatre connexités sur les pixels verts. Nous nous servons du canal bleu pour marquer les pixels déjà parcourus. Lorsque la région est complétée (plus de voisin vert) nous calculons le barycentre de la zone. Ensuite, nous continuons le parcours de l'image pour trouver de nouveaux noyaux, c'est à dire d'autres marqueurs verts. La taille des régions est contrainte par une valeur minimale. Cette valeur permet de supprimer les bruits dans l'image (tâches vertes de petite taille).

Lors de la **deuxième étape**, nous associons les sommets de l'interacteur aux marqueurs. L'utilisateur dispose toujours les interacteurs un à un dans le champ de la caméra. La détection de trois nouveaux marqueurs implique la présence d'un

nouvel interacteur sur le plateau.
Nous identifions les marqueurs sur les interacteurs toujours dans le même ordre et avec la même orientation en calculant les produits vectoriels et les valeurs des angles (cf. fig. 3.7). À chaque marqueurs nous associons une lettre a, b, c correspondante à la numérotation du sommet. Nous disposons les sommets de manière à ce que le résultat du produit vectoriel $\overrightarrow{bc} \wedge \overrightarrow{ba}$ ait la valeur de composante en $\overrightarrow{z}$ négative. Si la composante $\overrightarrow{z}$ est positive nous inversons a et c pour changer l'orientation. Cette technique nous permet de toujours identifier les sommets dans le même sens, en l'occurence le sens indirect. Ensuite, nous calculons les angles entre a, b et c pour identifier l'angle droit et permuter les lettres jusqu'à ce que b soit au sommet de l'angle droit. Pour trouver l'orientation de l'interacteur, nous calculons l'angle formé par la droite (bc) et l'axe des abscisses. La position du centre de l'interacteur est le milieu de l'hypoténuse. À ce stade nous connaissons l'orientation de l'interacteur et la position de son centre. Toutes ces informations sont conservées dans une structure pour l'image suivante.

Lors de la **troisième étape**, nous formulons l'hypothèse que les déplacements de chaque interacteur entre deux images capturées par la caméra sont faibles. Ainsi pour connaître la nouvelle position d'un interacteur nous détectons, dans un premier temps, la position de tous les marqueurs dans l'image, puis, dans un second temps, nous identifions les décalages des marqueurs à partir de la position des marqueurs dans l'image antérieure. Avec l'hypothèse précédente, nous cherchons les marqueurs les plus proches de l'ancienne position parmi la liste des nouveaux marqueurs. Si le nombre de marqueurs détectés est égal à trois fois le nombre d'interacteurs alors l'algorithme se termine. Si le nombre de marqueurs augmente de trois, nous considérons qu'il y a un nouvel interacteur dans le champ de la caméra. Dans ce cas, nous créons un nouvel interacteur dans notre programme et lui associons les trois nouveaux marqueurs. Si le nombre de marqueurs diminue, trois causes peuvent en être à l'origine. Les marqueurs peuvent être mis hors du champ de la caméra par l'utilisateur. Ils peuvent également disparaître de l'image à cause d'une occlusion ou d'un brusque changement de lumière. La dernière cause est la mise en contact de deux marqueurs, la caméra ne pouvant les différencier, notre algorithme détecte une seule et unique région pour les deux marqueurs. Nous différencions les deux premiers cas du troisième en calculant la surface de la région verte. Dans le cas d'un accroissement de région, 3^{ieme} cas, nous conservons l'ancienne position et la déplaçons légèrement vers le centre de la nouvelle région. Dans le cas de la perte d'un marqueur, nous conservons l'ancienne position.

L'utilisation de l'historique des images nous permet d'éviter le tri de l'ensemble des points à chaque nouvelle image et de lever des ambiguïtés sur la position des interacteurs. Cependant cette technique impose de capturer les images avec une cadence supérieure à dix images par seconde pour permettre ce suivi.

3.3.1.2 Discussion

Actuellement, nous arrivons à suivre cinq interacteurs (fig. 3.8) en temps réel avec une cadence supérieure à 15 images par seconde. Nous récupérons les positions (x,y) et la rotation autour de l'axe $\overrightarrow{oz}$ de chaque interacteur. Cet algorithme, intégré dans le logiciel de SKUA, nous permet une interaction en temps réel malgré l'affichage de modèles CAO. Nous avons réalisé des tests sur deux types d'ordinateur et nous maintenons une cadence supérieure à dix images par seconde. Le premier ordinateur est équipé d'un pentium 4 cadencé à 2,8 Ghz avec 1 giga Octet de mémoire vive et d'une carte graphique GeforceFx 5900. Le second un ordinateur est un ordinateur portable équipé d'un pentium M cadencé à 1,7 Ghz avec 512 Mo de mémoire vive et une carte graphique Geforce 5250 Go. Sur les deux configurations nous obtenons des performances supérieures à dix images par seconde.
Cette méthode de suivi présente cependant deux contraintes à cause de l'utilisation des marqueurs et d'une seule caméra. La première contrainte est l'utilisation des interacteurs dans le plan. La plupart des utilisateurs aimeraient mouvoir les interacteurs dans l'espace 3D. La seconde contrainte apparaît lorsque des marqueurs se touchent. Enfin, si l'utilisateur masque et déplace en même les marqueurs, notre algorithme de suivi mélange alors les marqueurs.

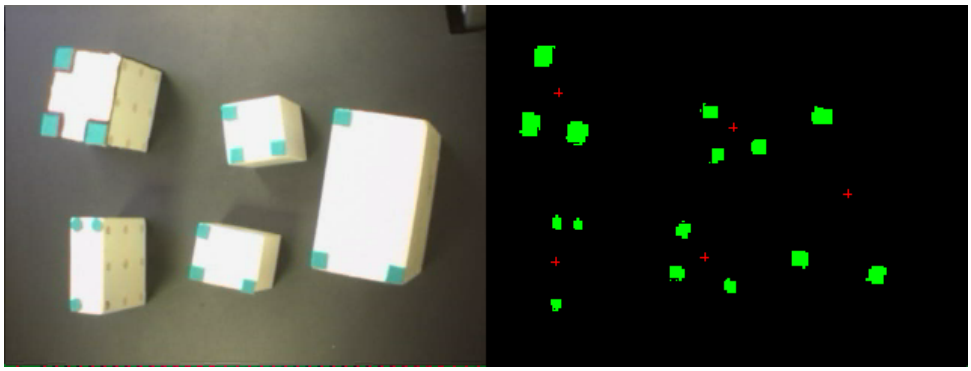


Fig. 3.8 – Exemple de suivi d'interacteurs.

3.3.2 Approche basée sur des modèles 3D

Les travaux récents utilisant l'approche par modèle 3D nous lever la contrainte du suivi des interacteurs dans le plan. En effet, avec une seule caméra il est possible de trouver la position et l'orientation d'un objet en connaissant la géométrie des objets suivis et les propriétés de la caméra.
Suivant cette approche 3D, les travaux de Leproux et Guitton [dlRG03], concernant la détection des mouvements de la main, et ceux de Hasenfratz, Lapierre, Gascuel et Boyer [HLGB03] sur la détection de la posture du corps humain aux-

quels nous nous sommes référés s'appliquent à des objets déformables. Dans notre cas, les objets suivis sont non déformables, à l'instar des travaux de Brown, Drummond et Cipolla [DC00] où ils suivent un objet rigide en temps réel. Ces travaux reposent sur une approche liée aux algèbres de Lie. Nous avons choisi pour notre part de nous appuyer sur une technique incluant le calibrage de la caméra. La calibration de la caméra, nous permet de diminuer le nombre d'inconnues. En effet, nous connaissons les déformations dues à la caméra, les seules inconnues dans ce cas sont les positions x,y et z de l'objet et ses orientations autour des axes.

Le principe des algorithmes par modèle est de comparer un modèle réel et un modèle virtuel (cf. fig. 3.9). Le modèle virtuel est obtenu à partir d'une caméra virtuelle simulant la caméra réelle. Ensuite, les images réelle et virtuelle sont comparées pour estimer les déplacements entre le modèle réel et le modèle virtuel. Nous avons décomposé l'algorithme en trois étapes : l'extraction des paramètres de la projection, la construction de l'image virtuelle et l'évaluation des déplacements.

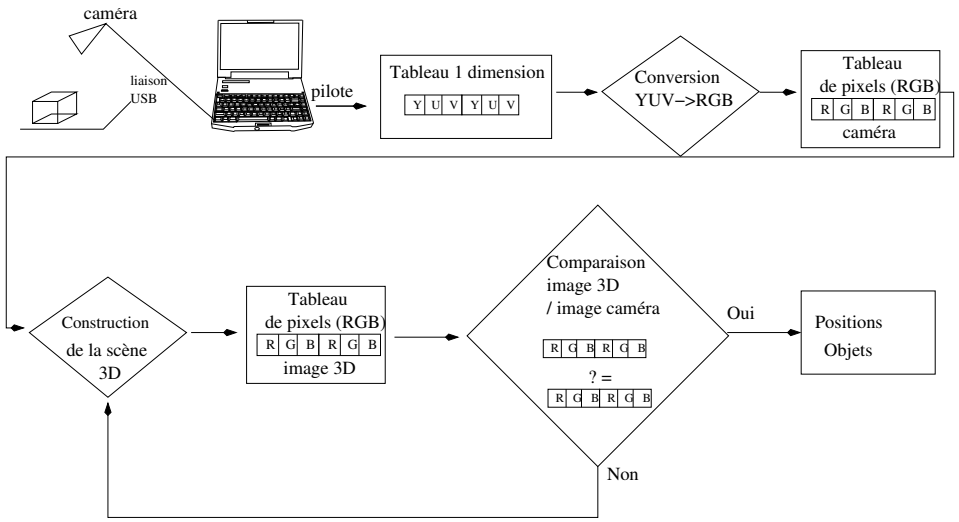


FIG. 3.9 – Processus global de l'approche par modèle.

Dans les prochaines sections, après avoir rappelé le principe de la formation d'une image, nous présentons nos travaux de calibrage d'une caméra ainsi que la manière de simuler une caméra avec OpenGL. Nous décrivons ensuite notre approche par modèle 3D permettant le suivi des interacteurs munis de marqueurs. Nous terminons par une critique de notre approche en résumant ses avantages et ses limitations.

3.3.2.1 Principe de la formation d'une image et de la calibration

La formation d'une image par une caméra (fig. 3.10, p 61) peut être assimilée à une transformation d'un espace 3D à un espace 2D sous forme d'une matrice

d'homographie. En termes projectifs, nous estimons que la matrice d'homographie représente la projection de la scène 3D en une image 2D. Cette matrice est caractérisée par les propriétés du système optique et la position relative du repère des objets et du repère de la caméra. Pour la rédaction de cette partie, nous nous sommes appuyés sur les travaux de Horaud et Monga [HM95].

Définition 7 *Le procédé permettant de calculer la position et les caractéristiques d'un système photographique (type caméra) relativement à une scène est appellé le* ***calibrage****. Ce procédé consiste à calculer la matrice d'homographie M_g. M_g est appelée matrice de projection perspective. Elle s'écrit généralement sous la forme suivante :*

$$M_g = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

De cette manière, tous les pixels de l'image sont le résultat de la projection de la scène réelle sur le plan image. Pour décrire notre système de caméra, nous ignorons les distorsions du système optique car nous avons remarqué à l'usage qu'elles sont quasi-inexistantes dans la zone de traitement. Dans cette partie nous caractérisons le modèle géométrique de notre système optique à partir de la matrice de projection perspective.

Notation 2 *Soit $(x_i, y_i, z_i, 1)^T$ les coordonnées homogènes des points de la scène dans le repère objet avec i, un entier positif. Nous notons $PR_{i/Ro}$ ces points (P) réels (R) dans le repère objet (Ro). $(x_i, y_i, z_i)^T$ sont les coordonnées cartésiens de ces points.*

Notation 3 *Soit $(su_i, sv_i, s)^T$ les coordonnées homogènes des points dans le repère image avec i entier positif. Nous notons $PI_{i/Ri}$ ces points (P) images (I) dans le repère image (Ri). Chaque point $PI_{i/Ri}$ est la projection du point $PR_{i/Ro}$. $(u_i, v_i)^T$ sont les coordonnées cartésiens de ces points dans le repère image.*

Les transformations de la caméra pour construire une image sont définies par $PI_{iR_i} = M_g.PR_{i/R_o}$ soit de manière plus détaillée :

$$\begin{pmatrix} su_i \\ sv_i \\ s \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_x \\ m_{21} & m_{22} & m_{23} & m_y \\ m_{31} & m_{32} & m_{33} & m_z \end{bmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{3.1}$$

D'après l'équation 3.1, nous pouvons écrire :

$$u_i = \frac{m_{11}x_i + m_{12}y_i + m_{13}z_i + m_{14}}{m_{31}x_i + m_{32}y_i + m_{33}z_i + m_{34}} \tag{3.2}$$

$$v_i = \frac{m_{21}x_i + m_{22}x_i + m_{23}z_i + m_{24}}{m_{31}x_i + m_{32}y_i + m_{33}z_i + m_{34}} \tag{3.3}$$

Que nous pouvons transformer en équations linéaires :

$$m_{11}x_i + m_{12}y_i + m_{13}z_i + m_{14} - u_i.m_{31}x_i - u_i.m_{32}y_i - u_i.m_{33}z_i = u_i.m_{34} \tag{3.4}$$

$$m_{21}x_i + m_{22}x_i + m_{23}z_i + m_{24} - v_i.m_{31}x_i - v_i.m_{32}y_i + v_i.m_{33}z_i = v_i.m_{34} \tag{3.5}$$

Des équations 3.4 et 3.5, nous pouvons écrire le système matriciel suivant :

$$\begin{bmatrix} & & \vdots & & & & \vdots & & & \vdots & \\ x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i.x_i & -u_i.y_i & -u_i.z_i \\ & & & & & & & & & & \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i.x_i & -v_i.y_i & -v_i.z_i \\ & & \vdots & & & & \vdots & & & \vdots & \end{bmatrix} \cdot \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} = \begin{pmatrix} u_i.m_{34} \\ v_i.m_{34} \end{pmatrix} \tag{3.6}$$

Pour calculer la matrice il faut poser le système d'équation 3.6 pour les points connus de la scène en déterminant les points $PI_{i/Ri}$ correspondants aux points $PR_{i/Ro}$. Ensuite, il faut calculer les coefficients de la matrice de projection perspective ; pour cette étape nous utilisons la bibliothèque Gandalf que nous détaillons par la suite.

3.3.2.2 Calibration de la caméra

Pour la calibration, nous proposons un système non contraignant et nécessitant peu d'interventions de l'utilisateur, car cette étape est nécessaire à chaque déplacement de la plate-forme. La difficulté réside principalement à mettre en correspondance les points de la scène réelle et leurs images dans l'image 2D an automatisant le plus au maximun les tâches.
Nous avons conduit une analyse de l'existant permettant de calibrer une caméra. Nous avons étudié deux bibliothèques en particulier : la bibliothèque OpenCV et

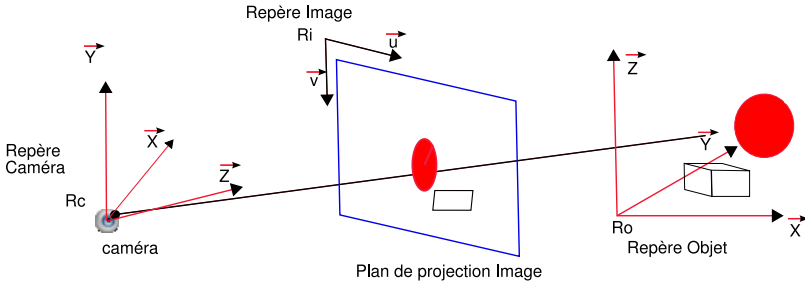


FIG. 3.10 – Schématisation d'un système optique.

la bibliothèque Gandalf. OpenCV Library [Opea],[Cor03](Open Source Computer Vision) et Gandalf [Gan], [McL03] sont deux bibliothèques C de traitement d'image sous licence LGPL.
Dans les deux prochaines parties, nous décrivons notre étude de ces deux bibliothèques au travers des résultats et notre choix d'utilisation, pour terminer par décrire le travail réalisé pour obtenir un algorithme de calibration correspondant aux critères de la plate-forme.

Calibration avec OpenCV

L'évaluation de la bibliothèque OpenCV Library [Opea],[Cor03](Open Source Computer Vision Library) fut motivée par le fait que nous voulions une solution performante et fiable. OpenCV est une bibliothèque C développée par Intel pour le traitement d'images. Elle fonctionne avec les systèmes d'exploitation Linux et Windows. Il existe un programme "CalibFilter" sous Windows qui permet de calibrer une caméra en utilisant un simple damier noir et blanc. Afin de déterminer les coefficients de la matrice intrinsèque, il faut capturer au minimum trois images valides du damier. Il est nécessaire que les positions du damier diffèrent entre chaque prise de vue. En pratique, nous devions utiliser entre cinq et huit images valables du damier, sinon les résultats pouvaient être erronés. Toutes ces étapes sont réalisées à l'aide des fonctions d'OpenCV. Nous avons réalisé le portage du programme "CalibFilter" de Windows sous Linux afin de pouvoir l'intégrer à notre application.
Dans notre configuration d'utilisation (caméra située à 1 m au dessus de la table et champ de vision de 60cm x 60cm) deux problèmes sont apparus : le temps de calibration et la validité des résultats. Le champ de vision réduit et la faible distance entre la caméra et le plateau ne permettent de déplacer suffisament la mire dans l'espace, ce qui entraîne souvent des calibrations longues de l'ordre de deux minutes. Les calibrations sont également contraignantes pour l'utilisateur car il doit en permanence bouger la mire en restant dans le champ de la caméra. Enfin, suite à une série de tests, nous avons remarqué que les résultats retournés

sont parfois erronés. En effet, la valeur calculée avec OpenCV de la focale de notre caméra varie entre 500 et 1112. Nous avons conclu que le problème venait de la position de la grille par rapport à la caméra, car dans notre configuration nous ne pouvions pas fournir de prises de vue suffisamment variées.
Dans notre configuration, OpenCV ne nous a pas apporté satisfaction, nous avons donc évalué les possibilités de la bibliothèque Gandalf.

Calibration avec Gandalf

Précédemment nous avons décrit la calibration comme le calcul de la matrice de projection perspective transformant la scène en une image. Le processus global de la calibration est décrit par le schéma 3.11.
Pour calculer cette matrice de projection perspective avec la fonction de Gandalf (*gan_ homog34_ solve*), nous devons reconnaître au moins six points dans l'image ($PI_{i/Ri}$) et connaître leurs coordonnées ($PR_{i/Ro}$) dans la scène.
Pour réaliser les différentes étapes, le calcul de la matrice et de reconnaissance des points dans l'image, nous avons utilisé les fonctions disponibles dans la bibliothèque Gandalf.

Pour détecter les points $PI_{i/Ri}$ dans l'image nous avons essayé les filtres de détection de la bibliothèque, décrits par la suite, que nous avons jugé pertinents. Une fois les points $PI_{i/Ri}$ et $PR_{i/Ro}$ mis en correspondance, nous avons calculé la matrice M_g en utilisant la fonction *gan_ homog34_ solve* de Gandalf. Cette méthode retourne toujours une matrice en approximant la solution.

Notation 4 *Nous notons $M_{g'}$ la matrice calculée par la fonction de Gandalf.*

Comme il s'agit d'une approximation, nous devons évaluer l'erreur de projection entre la matrice $M_{g'}$ calculée par Gandalf et le système projectif de la caméra.

Notation 5 *Soit $(x'_i, y'_i, w'_i)^T$ les coordonnées homogènes de la projection des points $PR_{i/Ro}$ dans le repère image par la matrice M'_g tel que :*

$$(x'_i, y'_i, w'_i)^T = M'_g \cdot PR_i$$

Rappel : $(u_i, v_i)^T$ sont les coordonnées des points dans le repère image avec i entier positif de la projection des $PR_{i/Ro}$.

Définition 8 *Soit ε l'erreur moyenne entre les points que nous avons identifiés dans l'image et la projection de ces mêmes points par la matrice M'_g :*

$$\varepsilon = \frac{\sum_{i=0}^{n} (\frac{x'_i}{w'_i} - u_i) + (\frac{y'_i}{w'_i} - v_i)}{n};$$

Pour permettre une calibration en limitant l'intervention de l'utilisateur nous devons identifier de manière automatique les pixels $PI_{i/Ri}$ de l'image correspondants aux points $PR_{i/Ro}$ dans l'espace objet. Pour la détection, nous avons essayé plusieurs techniques de traitement d'images en nous appuyant sur deux types de cibles : l'une utilise des marqueurs, l'autre repose sur l'utilisation d'un damier.

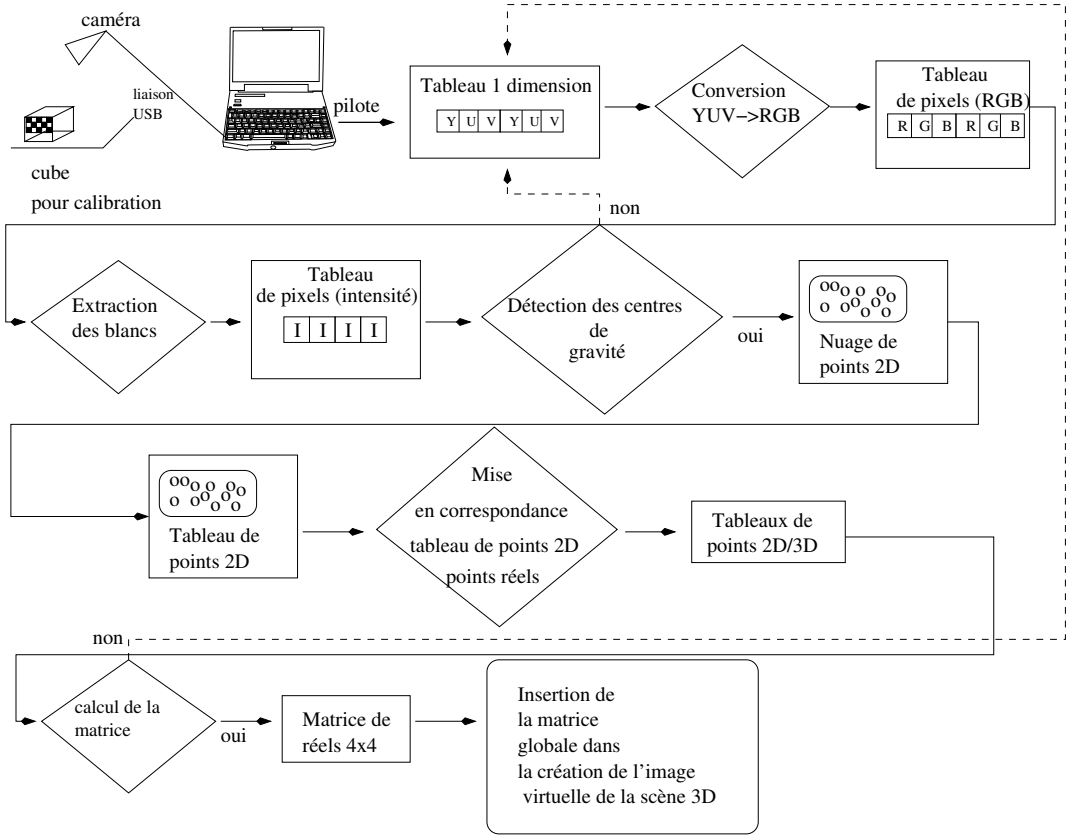


FIG. 3.11 – Description du processus de calibration.

Cible : les marqueurs

La première solution que nous avons essayée est d'utiliser des marqueurs noirs de forme carrée comme cible. La couleur noire permettait de limiter les problèmes de contraste et de luminosité liés au capteur de la caméra. Ces marqueurs (fig. 3.12) étaient disposés à plusieurs endroits de la scène sur des interacteurs et sur la table.

Nous avions implémenté un algorithme pour détecter les marqueurs en nous appuyant sur le filtre de Harris ([HS88]) de la bibliothèque Gandalf. Cette fonction retourne les coordonnées des sommets des angles présents dans l'image comme l'illustre la figure 3.12. À partir de cette liste de points, nous avions réalisé un algorithme (cf. algo. 1,p. 64) pour les trier et détecter les centres des carrés :

```
p1, p2, p3, p4, p5, i, : entier ;
permut : booléen ;
L : liste de points
Pcentre : point
p1=0 ;p2=1 ;p3=2 ;p4=3 ;

Répéter
    Répéter
        permut=false
        Pour i de 1 à 4 faire
            Si (dist(L[p1],L[i]) < dist(L[p1],L[i+1])) Alors
                inverse(L,i,i+1) ; permut=true ;
            Fin Si
        Fin Pour
    jusqu'à ce que (permut)
    Répéter
        permut=false
        Pour i de 5 à taille(L) faire
            Si (dist(L[p1],L[i]) < dist(L[p1],L[p4])) Alors
                inverse(L,i,i+1) ; permut=true ;
            Sinon
                Si (dist(L[p1],L[i]) < dist(L[p1],L[p4])) Alors
                    inverse(L,i,i+1) ; permut=true ;
                Sinon
                    Si (dist(L[p1],L[i]) < dist(L[p1],L[p4])) Alors
                        inverse(L,i,i+1) ; permut=true ;
                    Fin Si
                Fin Si
            Fin Si
        Fin Pour
    jusqu'à ce que (permut)
    Si (fasb(dist(L[p1],L[p2]) - dist(L[p3],L[4])) < ε) Alors
        Pcentre=centre4points(L) ;p5=4 ;
    Sinon
        Pcentre=centre3points(L) ;p5=3
    Fin Si
    Si (Pcentre.x > 0) Alors
        supprimerNbElt(L,p5) ;
        stocker(Pcentre) ;
    Sinon
        supprimerNbElt(L,1)
    Fin Si

jusqu'à ce que (taille(L) > 3)
```

Algorithme 1: Extraction du centre des carrés

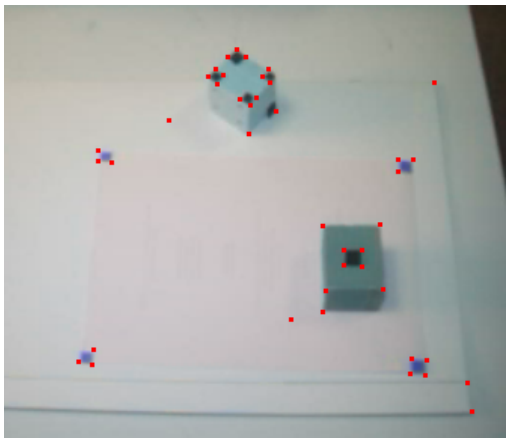


FIG. 3.12 – Résultat après le filtre de Harris (carré rouge).

Nous cherchons les quatre points les plus proches et testons s'il s'agit d'un carré en calculant la distance de chaque coté. Si le test est négatif, nous vérifions si ces trois premiers points forment la base d'un carré en calculant l'angle et les distances entre les trois points (cf. fig. 3.13).

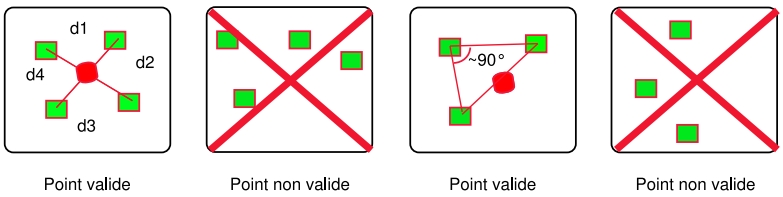


FIG. 3.13 – Test de l'algorithme des carrés.

Grâce à l'algorithme 1 nous détections les centres de chaque marqueur (cf. fig 3.14) avec une précision satisfaisante. En effet, en comparant la position des centres calculés par l'algorithme avec la position que nous pensions exacte nous estimions l'erreur moyenne inférieure à trois ou quatre pixels, ce que nous jugions satisfaisant en précision. De plus, le temps de calcul nécessaire était faible car nous arrivions à maintenir la cadence de la webcam à 15 images par seconde. Nous pouvions donc conclure que la position estimée des cibles, $PI_{i/Ri}$, dans l'image était juste.
À partir de ces cibles $PI_{i/Ri}$ et des marqueurs nous calculions la matrice de projection perspective M'_g avec la fonction de Gandalf. Pour cette matrice, ε (cf. def. 8, p 62), était inférieure à deux pixels, ce que nous jugions suffisant pour le suivi que nous voulions réaliser. Pour tester la justesse de la matrice de projection perspective nous avions calculé l'image de points n'appartenant pas à la cible. Nous avions pris une série de points, dont nous connaissions la position, à différents endroits dans la scène, c'est à dire sur le sol, aux bords de la scène et en hauteur.

Ensuite, nous avions fait une capture de l'image de la scène pour déterminer la position de l'image de ces points dans le repère image. Puis, pour évaluer l'erreur, nous avions calculé les positions des points de la scène avec la matrice M'_g. Pour les points possédant une forte élévation, nous trouvions entre les points trouvés dans l'image et les points calculés une différence supérieure à cinq pixels.

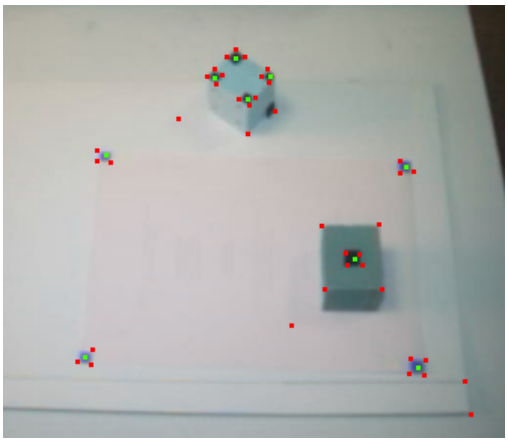


FIG. 3.14 – Résultat après la détection des centres (carrés verts).

Nous en avions conclu que cette erreur était due au faible nombre de points et que nous avions une interpolation trop faible. Nous nous sommes donc inspirés de la calibration avec OpenCV en utilisant une mire offrant un plus grand nombre de points.

Cible : damier

La deuxième solution envisagée comme cible fut un simple damier noir et blanc identique à celui d'OpenCV.
Notre algorithme basé sur le filtre de Harris n'était plus efficace avec cette mire. Nous obtenions un nuage de points trop dense (cf fig. 3.15) et non-ordonné. Nous ne parvenions pas à faire identifier rapidement les points PR de la mire dans l'image.

Nous avons alors cherché parmi les filtres disponibles dans Gandalf ceux répondant à notre problème. Nous avions essayé le filtre de détection de segments de droites basé sur l'algorithme de Canny [Can86]. Grâce aux équations de droite nous pensions reconstruire le damier pour en extraire les points caractéristiques.

Les résultats des équations de droite n'étaient pas suffisamment précis pour détecter la position de chaque carré du damier (cf fig. 3.16). Les deux causes identifiées étaient la faible taille des carrés du damier et la qualité du rendu des couleurs de notre système optique.
Après avoir testé les deux bibliothèques, OpenCV et Gandalf, sans succès, nous

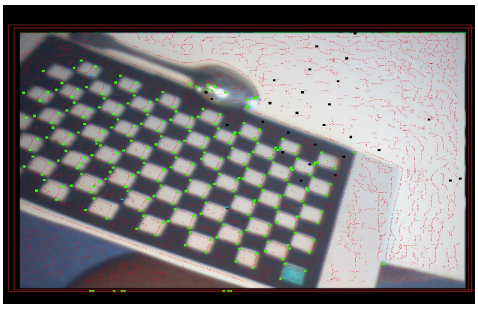


FIG. 3.15 – Résultat du filtre de Harris sur le damier.

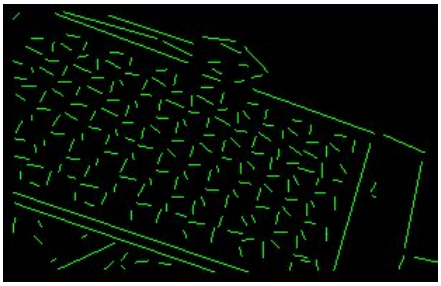


FIG. 3.16 – Résultat du filtre de Canny sur le damier.

avons décidé de développer notre propre algorithme de calibration pour la détection de la cible.

Calibration avec notre algorithme

Suite aux problèmes rencontrés auparavant et à la spécificité de notre mire de calibration nous avons décidé de développer un algorithme détectant (cf fig. 3.18) le barycentre des carrés. L'algorithme se décompose en trois phases :

- extraction de la couleur blanche ;
- calcul du barycentre des segments blancs pour chaque ligne horizontale de l'image et conservation des barycentres ayant une valeur proche de la largeur des carrés ;
- calcul de l'alignement des barycentres et de leurs distances pour vérifier que la forme est un carré.

Nous obtenons une image de la caméra au format RVB[4] stockée dans un tableau à une dimension de type "unsigned char".

Remarque 1 *Pour le traitement des couleurs, nous sommes passés dans l'espace TSL*[5] *pour éliminer les problèmes de contraste. Le passage en TSL prenant des*

[4] RVB : Rouge Vert Bleu.
[5] TSL :Teinte Saturation Luminance

ressources machines, nous avons décidé de comparer la qualité des résultats de l'extraction de couleur entre les espaces RVB et TSL. Pour l'extraction de couleurs simples (rouge, vert, bleu, blanc) nous obtenons des résultats quasi-identiques avec notre modèle de caméra. Nous travaillons donc systématiquement dans l'espace RVB pour un gain en ressources machines.

La **phase 1** est une simple extraction de la couleur blanche ; pour cela nous utilisons l'algorithme 2.

Si ((0.29 . R + 0.58 . V + 0.13 . B) > seuil) **Alors**
  | blanc
Sinon
  | noir
Fin Si

Algorithme 2: Filtre d'extraction des blancs.

L'utilisateur peut agir sur la valeur "seuil" du filtre en fonction de la luminosité ambiante et de la qualité de la caméra.

Lors de la **phase 2** (cf. algo. 3.3.2.2) nous calculons le barycentre de chaque ligne blanche et calculons son poids. Nous définissons son poids comme la somme de la longueur du segment et de la valeur 150.

Remarque 2 *L'ajout de la valeur 150 nous permet de rendre visible tous les barycentres même ceux dont le segment est inférieure à 20 pixels.*

Nous considérons que le barycentre est valable si la valeur du poids est comprise entre deux valeurs déterminées heuristiquement mais modifiables avec l'interface. Cela signifie qu'il peut appartenir à un carré.

```
x, y, deb, fin, erreur, ylargeur : entier ;

Pour y de 0 à hauteurImage faire
    ylargeur=y*largeur ;
    Pour x de 0 à largeurImage faire
        Si (Image[x+ylargeur] == blanc) Alors
            deb = fin =x ;
            erreur = 0 ;
            Répéter
                Si (Image[x+ylargeur] == blanc) Alors
                    fin=x ;
                    Image[ylargeur + x] = 30 ;
                Sinon
                    erreur++ ;
                Fin Si
                x++ ;
            jusqu'à ce que (erreur < 3 et x < largeurImage)
            Si (fin-deb > TailleCarreMini et fin-deb < TailleCarreMaxi)
            Alors
                Image[ylargeur + fin/2 + deb] = 150+fin ;
            Fin Si
        Fin Si
    Fin Pour
Fin Pour
```

Algorithme 3: Extraction des barycentres horizontaux.

Dans la troisième **phase**, nous calculons le centre des segments continus formés par les barycentres extraits à partir de l'algorithme 3. Nous vérifions également durant cette phase que les lignes ne soient pas trop courbées et que leurs tailles soient comprises entre la taille maximale et la taille minimale des carrés. L'algorithme est illustré par le schéma 3.17, et la photographie 3.18 illustre le résultat.

L'enchaînement des trois phases permet de détecter les centres des carrés avec une précision suffisante pour la calibration même lorsque la grille est penchée jusqu'à un angle de 45°. La valeur ε est inférieure à 2,5.
Cependant, nous avions une erreur importante lorsque les points réels avaient une valeur supérieure à 15cm suivant la coordonnée z. Pour résoudre les erreurs d'imprécision du calcul de la matrice de projection perspective, nous nous sommes orientés vers l'utilisation d'une grille de calibrage qui devait posséder des points dans tout l'espace. Nous avons donc fabriqué un support cubique pour la mire de calibrage (cf. fig. 3.19, p. 71) de 200 mm x 200mm x 200mm. Il s'agit d'un cube en bois recouvert d'une peinture noire mate pour diminuer les reflets dont deux faces perpendiculaires sont recouvertes de damiers rouge et blanc. Nous obtenons un ensemble de points situés dans l'espace 3D sur les plans (y,x) et (x,z).

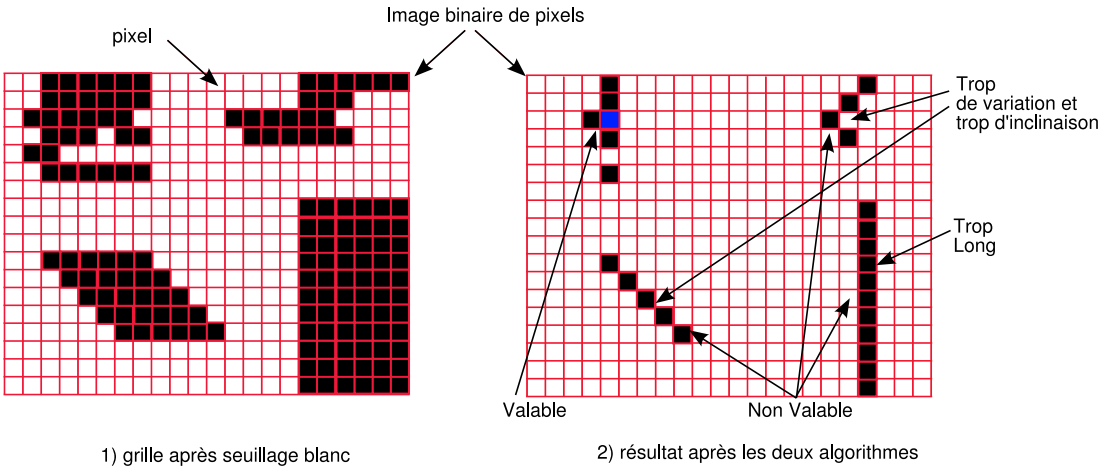


FIG. 3.17 – Illustration des algorithmes de détection des barycentres.

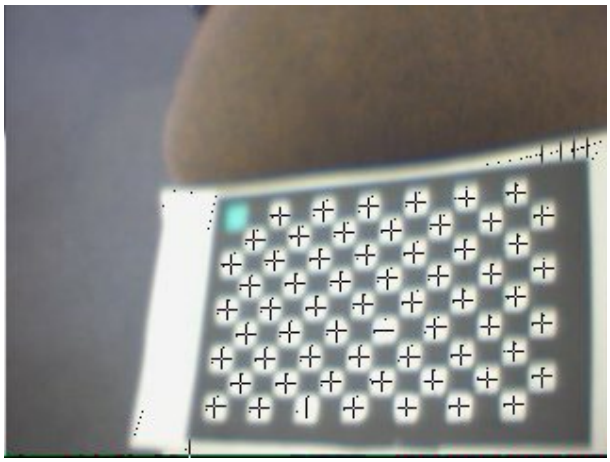


FIG. 3.18 – Résultat de l'algorithme par calcul des barycentres.

Nous calculons la position des points du damier blanc puis ceux du damier rouge. Pour détecter les barycentres des carrés, nous exécutons les trois phases décrites précédemment avec un filtre d'extraction des blancs et des rouges lors de la phase 1. Ainsi, le double enchaînement des trois phases permet de calibrer tout l'espace sans aucune intervention de l'utilisateur.

Pour améliorer la précision d'extraction des barycentres nous avons optimisé nos algorithmes sur deux critères. Le premièr est l'élimination des objets de petite dimension et la séparation des formes dans l'image après le seuillage (phase 1). Nous appliquons successivement deux filtres d'érosion puis un filtre de dilatation. L'application des quatre masques linéaires symétriques (deux diagonaux, un hori-

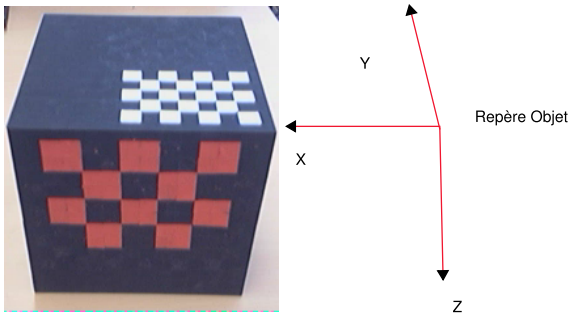


FIG. 3.19 – Mire 3D : le cube.

zontal et un vertical) retire les deux pixels les plus proches du bords quelques soit la direction. Cette technique permet d'affiner le contour des formes et de réduire les erreurs dues à la qualité de l'image (cf fig. 3.20). Elle peut s'apparenter à la réalisation d'un filtre de fermeture. En ce qui concerne le second critère, nous sa-

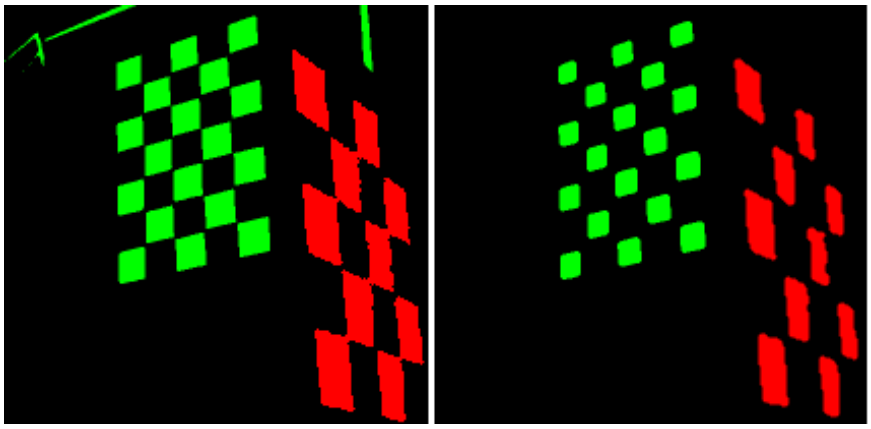


FIG. 3.20 – Résultat après application des différents filtres.

vions que les barycentres forment un damier régulier et que les systèmes optiques avec de faibles distorsions conservent l'alignement de points.
Donc, après avoir extrait les barycentres (phase 3), nous calculons les équations des droites approximant les barycentres qui sont, en théorie, alignés en utilisant la méthode des moindres carrés.

Ensuite nous calculons les coordonnées des intersections des droites pour estimer la position des barycentres. En effet, chaque barycentre est l'intersection de deux ou trois droites. Nous diminuons ainsi l'erreur en replaçant les points les uns par rapport aux autres. Nous répétons ceci (calcul des équations de droites et intersections) tant que la différence entre les points donnant les équations de droites et les nouveaux points calculés avec les intersections de droite est supérieure à une certaine valeur V. V a été trouvée de manière heuristique. Si l'algorithme diverge,

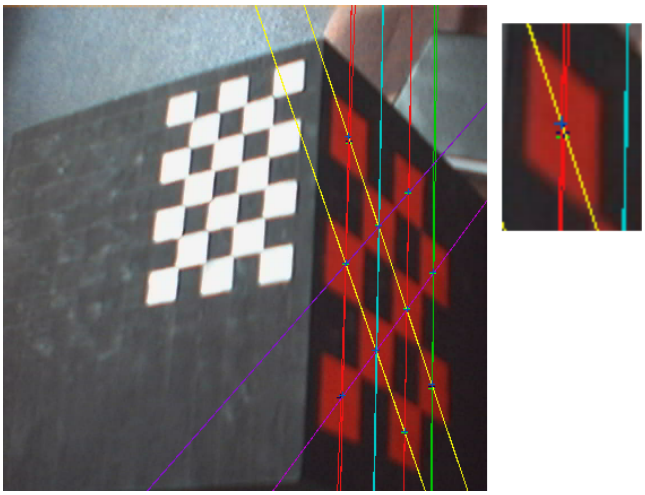


FIG. 3.21 – Résultat après application de l'algorithme des moindres carrés (source : points verts, résultat : points bleus.)

des points sont erronés et dans ce cas nous arrêtons le calcul et recommençons les calculs sur l'image suivante. Dans le cas contraire, les points sont validés (cf fig. 3.21).

Après avoir déterminé tous les coordonnées des barycentres, nous utilisons la bibliothèque Gandalf pour calculer la matrice de projection perspective. Nous vérifions la validité de la matrice en évaluant ε (cf. def. 8, p 62).
Avec une résolution de caméra de 320x240 pixels, nous avons une erreur ε proche de 1 pixel (0.80). Nous sommes parvenus à réaliser une calibration quasi automatique. L'utilisateur n'intervenant que pour changer les valeurs des filtres (seuil des couleurs, et taille des carrés) suivant les conditions de luminosité.

Sauvegarde XML des paramètres de la caméra

Pour simplifier l'utilisation de SKUA nous utilisons un système de sauvegarde des paramètres de notre calibration dans un fichier au format XML (eXtensible Markup Langage). Comme ce fichier est représentatif de la simulation nous avons décidé de calculer les paramètres intrinsèque et extrinsèque de la calibration. La matrice intrinsèque caractérise les paramètres physiques de la caméra. La matrice extrinsèque caractérise les paramètres de transformation entre le repère objet et la repère caméra.
Nous calculons ces deux matrices à partir de la matrice de projection perspective décrite dans les travaux par Horaud et Monga dans [HM95].

Rappel : La matrice de projection perspective M_g s'écrit de la manière sui-

vante :

$$M_g = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} m_1 & m_{14} \\ m_2 & m_{24} \\ m_3 & m_{34} \\ m_4 & m_{44} \end{bmatrix}$$

où pour simplifier nous notons $m_i \Leftrightarrow \begin{bmatrix} m_{i1} & m_{i2} & m_{i3} \end{bmatrix}$;

Propriété 1 *Houraud définit la matrice intrinsèque, M_{int}, de la manière suivante :*

$$M_{int} = \begin{bmatrix} \alpha_u & 0 & u_o & 0 \\ 0 & \alpha_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

où :

$$u_0 = m_1.m_3 \qquad ; \qquad v_0 = m_2.m_3$$

$$\alpha_u = -\|m_1 \wedge m_3\| \qquad ; \qquad \alpha_v = \|m_2 \wedge m_3\|$$

Remarque 3 *Tous ces paramètres sont caractéristiques de la caméra.*
u_0, v_0 sont les produits des coordonnées du centre optique dans le repère image (mesurées en pixels) et de la distance focale de la caméra. α_u, α_u sont respectivement le produit des facteurs d'échelle vertical et horizontal ,exprimés en pixels par millimètre, et de la distance focale.

Propriété 2 *Houraud définit la matrice extrinsèque, M_{ext}, de la manière suivante :*

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

où d'après [HM95] :

$$\begin{cases} r_1 &= \frac{1}{\alpha_u}(m_1 - u_0 * m_3) \\ r_2 &= \frac{1}{\alpha_v}(m_2 - v_0 * m_3) \\ r_3 &= m_3 \\ t_x &= \frac{1}{\alpha_u}(m_{14} - u_0 * m_{34}) \\ t_y &= \frac{1}{\alpha_v}(m_{24} - v_0 * m_{34}) \\ t_z &= m_{34} \end{cases}$$

Pour stocker tous ces paramètres nous avons créé notre propre système de balise basé sur le langage XML.

Tous les paramètres stockés sont mis entre les balises générales <Camera> et </Camera>. Puis nous stockons le modèle de la caméra entre les balises <Camera :Marque> </Camera :Marque> sous la forme d'une chaîne de caractères.

```
<Camera :Camera>
<Camera :Marque>
Philips 740 webcam </Camera :Marque>
<Camera :MGlobale>
<Camera :xx> -2,943774 </Camera :xx>
<Camera :xy> 0,206043 </Camera :xy>
......          ......          ......
<Camera :zy> -0,000756 </Camera :zy>
<Camera :zz> -0,005784 </Camera :zz>
<Camera :zw> -4,198821 </Camera :zw>
</Camera :MGlobale>
<Camera :MIntrinseque>
<Camera :xx> -0,742581 </Camera :xx>
<Camera :xy> 0,000000 </Camera :xy>
<Camera :xz> 0,182850 </Camera :xz>
......          ......          ......
</Camera :MIntrinseque>
<Camera :MExtrinseque>
<Camera :xx> -0,742581 </Camera :xx>
<Camera :xy> 0,000000 </Camera :xy>
<Camera :xz> 0,182850 </Camera :xz>
......          ......          ......
<Camera :wz> 0,000000 </Camera :wz>
<Camera :ww> 0,000000 </Camera :ww>
</Camera :MExtrinseque>
</Camera :Camera>
```

FIG. 3.22 – Exemple de notre format de fichier XML

La matrice globale de projection perspective est décrite entre les balises <Camera :MGlobale> et </Camera :MGlobale>, la matrice extrinsèque entre les balises <Camera :Mextrinseque> et </Camera :MExtrinseque>, et la matrice intrinsèque <Camera :Mintrinseque> et </Camera :MIntrinseque>.
Chaque coordonnée de la matrice est stockée en fonction de sa position dans la matrice à la manière de la numérotation de Gandalf. Les balises entourant chaque élément sont constituées du mot "Camera :" suivi de deux lettres identifiant la la ligne et la colonne de la position dans la matrice. La première lettre indique la ligne, soit x, y et z pour respectivement la première, deuxième et troisième lignes. La seconde lettre indique la colonne ou x indique la première colonne et y, z etw respectivement les seconde, troisième et quatrième colonnes.

3.3.2.3 Simulation d'une caméra réelle en OpenGL

Le modèle de caméra d'OpenGL est différent de celui d'une caméra réelle. La matrice de projection perspective que nous avons extraite transforme les points de la scène $PR_{i_{/Ro}}$ en pixels $PI_{i_{/Ri}}$ dans un plan image 2D (0,largeur,hauteur).

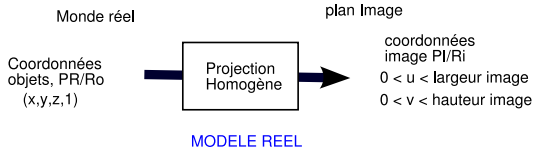


FIG. 3.23 – Illustration de la projection d'un modèle de caméra réelle.

En revanche, le pipeline graphique OpenGL projette les points $PR_{i_{/Ro}}$ dans un espace cubique borné entre $< 1,1,1 >$ et $< -1,-1,-1 >$. Les figures 3.23 et 3.24 décrivent respectivement les processus graphiques d'une caméra (modèle réel) et d'OpenGL.

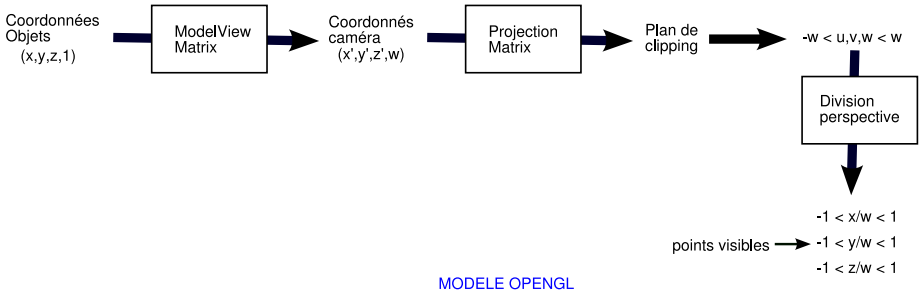


FIG. 3.24 – Illustration de la projection du modèle OpenGL.

Avec OpenGL, les coordonnées de $PI_{i_{/Ri}}$ sont de la forme (u, v, z_a) où u et v sont compris entre -1 et 1. Pour un modèle de caméra réelle, les coordonnées $PI_{i_{/Ri}}$ sont tels que u est positif et inférieur à la largeur de l'image et v est positif et inférieur à la hauteur de l'image. Pour faire correspondre les transformations entre le modèle réel et le modèle OpenGL, nous calculons la matrice de projection d'OpenGL à partir de la matrice de projection perspective M_g calculée précédemment :

Notation 6 *Nous notons M_{ogl} la matrice de projection dans le processus graphique OpenGL.*

Notation 7 *Nous notons lI la résolution horizontale de la camera lors du calibrage.*

Notation 8 *Nous notons hI la résolution verticale de la camera lors du calibrage.*

$$M'_{1,i} = \frac{2}{lI} M_{g_{1,i}} \; \forall i \mid 1 \leq i \leq 4 \tag{3.7}$$

$$M'_{2,i} = \frac{2}{hI} M_{g_{2,i}} \; \forall i \mid 1 \leq i \leq 4 \tag{3.8}$$

$$M'_{3,i} = 0 \; \forall i \in \aleph \mid 1 \leq i \leq 4 \; et \; M'_{3,3} = -M_{g_{3,3}} \tag{3.9}$$

$$M'_{4,i} = M_{g_{3,i}} \; \forall i \in \aleph \mid 1 \leq i \leq 4 \tag{3.10}$$

$$M_{ogl} = M'^{T} \tag{3.11}$$

La formule 3.7 déplace les valeurs de u entre 0 et 2. La formule 3.8 ramène également les valeurs de v entre 0 et 2. L'insertion de la troisième ligne de M_g dans la quatrième ligne de M' (formule 3.9 et 3.10) permet d'obtenir : -w $<$ u,v,z $<$ w . La formule 3.11 transforme la matrice dans le format OpenGL (transposé de la matrice).
Ceci nous permet d'avoir les points $PI_{i_{/Ri}}$ tel que : $0 < u, v, z_a < 2$. Ensuite, nous devons réaliser une translation de -1 des coordonnées u et v pour que la projection des points soit dans l'espace cubique [(-1,-1,-1),(1 1 1)]. Nous ne pouvons pas utiliser la fonction "glTranslatef" car elle agit dans l'espace 3D (sur $PR_{i_{/Ro}}$) et nous devons agir sur les coordonnées $PI_{i_{/Ri}}$. Nous avons programmé la carte 3D de manière à exécuter les translations avant les plans de coupe sur les coordonnées $PI_{i_{/Ri}}$. Nous réalisons cette opération à l'aide d'un programme "Vertex Shader" (cf. algo. 3.3.2.3, p77) stocké dans la carte graphique. Nous avons programmé en assembleur, car nous n'avons pas jugé nécessaire d'utiliser des langages de plus haut niveau tel cg de Nvidia ou HLSL d'OpenGL pour réaliser ce programme.

Nous obtenons des résultats satisfaisants observables sur le montage photographique fig. 3.26, page 78. L'image de droite est l'image filmée par la caméra, l'image de gauche est celle reconstruite en donnant la position du cube dans la scène. L'image du bas est la superposition des deux images (montage réalisé sous le logiciel GIMP). Grâce à ce photo-montage, nous constatons que la projection virtuelle est quasi identique à la projection réelle. Nous pouvons conclure que la matrice de projection perspective est juste et que notre modèle OpenGL de caméra est exact.

3.3.2.4 Discussion

Dans les paragraphes précédents nous avons décrit nos procédures de calibration et la manière de simuler une caméra réelle à partir de sa matrice de projection perspective en OpenGL.
Nous appelons l'image virtuelle l'image obtenue avec OpenGL par opposition à l'image réelle obtenue à partir de la caméra.

```
!!VP1.1
# Constants Registers :
# c[0]-c[3] = combined model-view-projection matrices
# c[4] = c[4].y = 1.0
# Input Registers :
# v[OPOS] = per-vertex position
# v[3] = per-vertex color
# Output Registers :
# o[HPOS] = homogeneous position
# o[COL0] = diffuse color
#Transform the per-vertex position by ModelViewProjection
DP4     R0.x, c[0], v[OPOS];
DP4     R0.y, c[1], v[OPOS];
DP4     R0.z, c[2], v[OPOS];
DP4     R0.w, c[3], v[OPOS];
#compute 1/w
RCP     R1.w , R0.w;
#Transform the per-vertex in uniform coordinate
MUL     R2.x , R0.x, R1.w;
MUL     R2.y , R0.y, R1.w;
MUL     R2.z , R0.z, R1.w;
MUL     R2.w , R0.w, R1.w;
#Translate to the center
SUB     R3.x, R2.x, c[4].y;
SUB     R3.y, R2.y, c[4].y;
#Copy the result to ouput register
MOV     o[HPOS].x, R3.x;
MOV     o[HPOS].y, R3.y;
MOV     o[HPOS].z, R2.z;
MOV     o[HPOS].w, R2.w;
MOV     o[COL0], v[3];
END
```

FIG. 3.25 – Programme “vertex shader” pour centrer l’image.

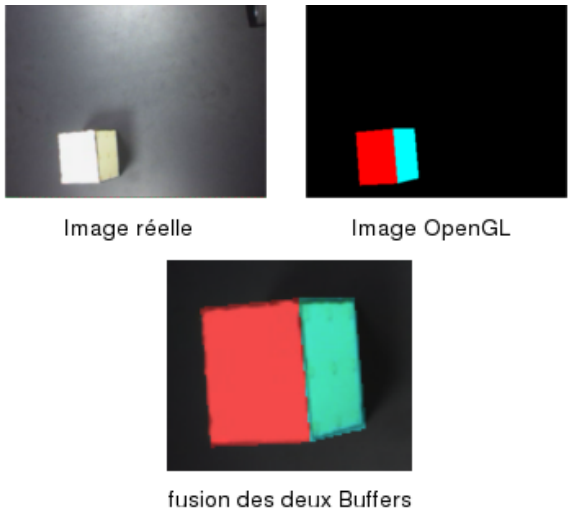


FIG. 3.26 – Résultat de la calibration et de la projection OpenGL.

Pour suivre l'objet (l'interacteur) nous comparons l'image virtuelle et l'image réelle pour déterminer les différences. Elles peuvent être calculées à partir des couleurs, des lignes, des points caractéristiques ou encore des enveloppes convexes.
L'utilisation des couleurs a été abandonnée à cause de la faible qualité et de l'instabilité des couleurs de notre caméra. Pour palier à ce problème, nous nous appuyons sur la détection de lignes en utilisant le filtre de Sobel (cf. fig. 3.27). Ces lignes correspondent aux arêtes des interacteurs. Pour déterminer la position des objets dans le plan, nous calculons le centre de gravité de chacun dans l'image virtuelle et dans l'image réelle. Les écarts de position entre les centres des objets réels et des objets virtuels déterminent le sens et la direction des translations à appliquer sur notre modèle 3D pour créer la nouvelle image virtuelle. Nous calculons le nouveau centre de gravité virtuel puis estimons le nouvel écart de position avec le centre de gravité réel. Nous réitérons ces quatre étapes (déplacement du modèle virtuel, création de l'image virtuelle, calcul du centre de gravité, estimation du déplacement) jusqu'à ce que les centres de gravité soient suffisamment proches.
Actuellement nous détectons la position (x,y) et essayons de déterminer l'orientation autour de l'axe $\overrightarrow{oz}$.
Les principales limitations de notre algorithme sont la difficulté à détecter précisément les couleurs pour trouver les zones à cause de la faible qualité de notre caméra (webCam philips proTouch). L'autre problème vient de l'utilisation de la carte 3D. Dans SKUA, nous affichons des modèles CAO complexes en utilisant la carte graphique. Nous pensons que la qualité du suivi par modèle 3D en pâtie. En effet, nous ne pouvons par créer assez d'images virtuelles, à cause du partage de ressources de la carte, pour estimer les déplacements en temps réel ($>$ 10 ips).
La dernière limitation est la mise en place des heuristiques pour détecter les dé-

placements des objets en 3D bloquée en partie par les deux problèmes précédents.

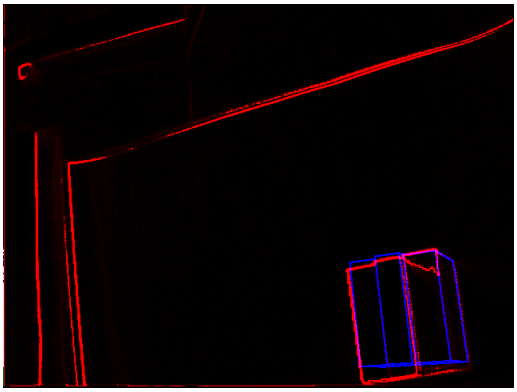


FIG. 3.27 – Superposition de la scène réelle (rouge ou gris clair) et de la scène virtuelle (bleu ou gris foncé).

3.3.3 Conclusion

Pour le suivi des interacteurs par vidéo nous avons proposé deux algorithmes différents.
Le premier algorithme repose sur une démarche classique de capture du mouvement qui consiste à suivre un point dans une image. L'idée ici est de considérer un marqueur comme un ensemble de trois points. Cette solution a l'avantage d'être peu coûteuse en temps de calcul ce qui nous assure une intégration efficace dans SKUA en permettant une visualisation temps réel des pièces CAO 3D coûteuses en CPU. Les limitations de cette technique sont au nombre de trois. Tout d'abord, elle n'offre pas un suivi en trois dimensions avec six degrés de liberté, mais seulement dans le plan avec trois degrés de liberté : deux translations et une rotation. Ensuite, le suivi d'un interacteur est perdu lorsque deux marqueurs sont en contact. L'algorithme regroupe les deux zones vertes en une seule et ne considère plus qu'un marqueur. Lorsque l'utilisateur déplace un interacteur en masquant les trois marqueurs il empêche le suivi : l'algorithme ne fonctionne plus, donc la pièce CAO associée n'est pas déplacée.

Pour résoudre ces limitations nous avons entrepris de mettre en place un algorithme reposant sur une approche par modèle en deux phases : la calibration de la caméra puis le suivi des interacteurs dans la scène.
Cette méthode nous permet de résoudre le problème liè au contact des interacteurs. En revanche, le problème d'occlusion avec la main lors des déplacements des interacteurs subsiste.
Notre implémentation de l'approche par modèle 3D s'est alors avérée trop coû-

teuse en temps de calcul pour permettre l'intégration. Nous étions très loin du temps réel nécessaire pour le suivi souhaité. Nous avions des résultats de l'ordre de 10 images par seconde pour le suivi d'un objet dans le plan. Cependant, cette dernière approche pourrait peut-être[6] devenir une solution satisfaisante en améliorant les heuristiques de prédiction et en adoptant d'autres solutions matérielles.

Une première solution matérielle-logicielle serait d'utiliser un rendu logiciel pour la création des images 3D nécessaires au suivi par modèle plutôt qu'un rendu matériel (solution actuelle). L'utilisation d'un rendu matériel nous oblige à recopier l'image 3D de la mémoire de la carte graphique vers la mémoire vive. Notre scène étant peu complexe, nous estimons que la somme du temps nécessaire pour créer la scène 3D avec la carte graphique et du temps de transfert vers la mémoire vive est supérieure au temps pour créer cette scène avec le processeur. Décharger la carte graphique des calculs de suivi nous permettrait donc d'avoir une meilleure fluidité pour l'affichage de nos modèles CAO et de créer plus d'images pour le suivi par modèle.

Une deuxième solution matérielle envisageable serait de changer de caméra. L'amélioration de notre caméra nous permettra d'avoir des couleurs et des contours plus nets, donc de réduire le bruit dans l'image et ainsi faciliter la mise en place des heuristiques. Nous avons acquis à cet effet une caméra analogique Sony XC-555 couplée avec une carte Matrox Chronos plus.
En résumé, le suivi d'interacteurs par vidéo nous semble aujourd'hui délicat à implémenter, en particulier pour un suivi lorsqu'il y a simultanément une occlusion et un mouvement ou lors de déplacement dans l'espace.

3.4 La plate-forme

Nous désirons réaliser une interface tangible opérationnelle. Afin d'obtenir le meilleur résultat possible, nous étudions en détail chaque composant (interacteur, plate-forme, communication interacteur-application) de notre I2HD.
Nous présentons dans cette partie la conception détaillée de la plate-forme en considérant l'environnement de travail et les contraintes ergonomiques.

La plate-forme est un support proposant un espace de manipulation pour l'utilisateur. Elle intègre la webcam qui est reliée au PC. Nous voulons proposer à l'utilisateur un espace de travail optimal prenant en compte les contraintes ergonomiques et facilitant l'utilisation des interacteurs.
Dans une première partie nous présentons notre étude préliminaire qui nous a conduit à la rédaction d'un cahier des charges, point de départ de la conception détaillée. Cette dernière est décrite dans la deuxième partie. Elle a conduit à une

[6] Nous n'avons pas eu le temps de tester ces solutions. Ceux deux pistes de recherche.

première solution [CCG+04] qui fut remise en cause par rapport à son coût et à sa solidité. Une nouvelle conception détaillée [VS04] a abouti à la solution finale.

3.4.1 Étude préliminaire

Lors de l'étude préliminaire nous avons établi un cahier des charges précisant les besoins et les contraintes liés à la plate-forme et à son utilisation.

Notre objectif principal pour cette plate-forme est d'offrir un environnement simple et pratique pour la manipulation des interacteurs. Dans son intégralité SKUA doit s'intégrer dans l'espace de travail sans le modifier (idéalement) et doit être une solution de travail individuel. Son coût de fabrication doit être inférieur à la valeur d'un PC. L'idée initiale de la plate-forme est illustrée par le photo-montage de la figure 3.28.

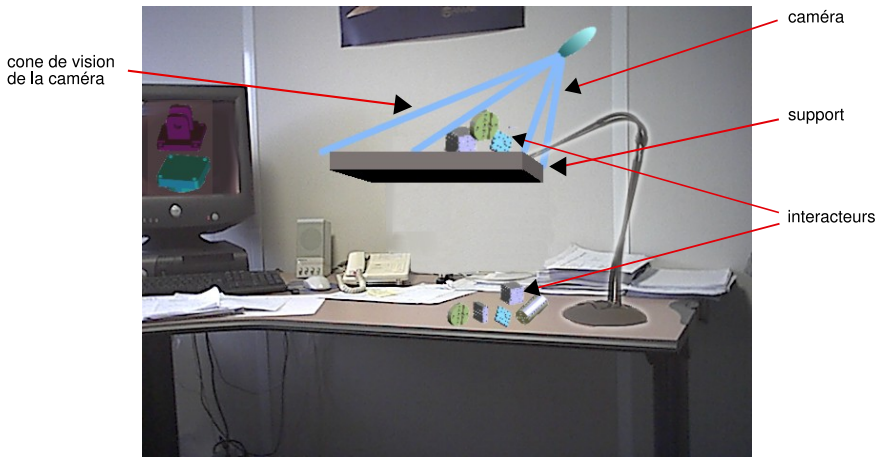


FIG. 3.28 – Représentation virtuelle et préliminaire de la plate-forme de SKUA.

Le plateau doit résister à l'effort exercé par la manipulation (appui des bras). La plate-forme doit pouvoir supporter une caméra à une hauteur variant entre 80 et 120 cm et un dispositif d'éclairage puissant de couleur blanche pour ne pas altérer la saisie des couleurs par la webcam. Le système de fixation de la caméra doit permettre une liaison linéaire (trois rotations et une translation) annulaire de la caméra par rapport à la plate-forme.
La plate-forme est principalement destinée à être utilisée au sein d'un bureau d'études. Le système doit s'insérer dans l'espace de travail (composé essentiellement d'un bureau, d'un ordinateur et d'un téléphone) sans modifier celui-ci. La plate-forme doit être amovible avec une position de rangement et occuper un encombrement minimal. Elle doit être accessible à tout moment, d'une mise en place aisée et permettre l'accès à la souris et au clavier ainsi que la visualisation

de l'écran en utilisation.

En partant de notre cahier des charges des étudiants de l'ESTIA (Axel Candeau, Florent Combellas, Peio Garcia, Bertrand Lajus, Nicolas Rey) ont proposé une solution lors de leur projet de fin d'étude que nous avons encadré. Tous les documents produits (le plan de développement, le plan de qualité, le cahier des charges fonctionnel, le cahier des charges marketing, le cahier des charges client) peuvent être obtenus sur simple demande par mail : l.garreau@estia.fr. Les paragraphes suivants synthétisent leur travail [CCG+04].

3.4.2 Contrainte ergonomique

La recherche de solutions tient compte des problèmes ergonomiques liés à la manipulation. Le système doit pouvoir être accessible par l'utilisateur à tout moment sans le contraindre dans son travail. Il doit pouvoir effectuer d'autres tâches tout en conservant le système en position de travail devant lui.
Suite à un entretien avec Patrick BADETS, ergonome du CRT ESTIA-Innovation, les étudiants ont dégagé les principales contraintes ergonomiques devant être prises en compte :

- l'utilisation de la plate-forme ne doit pas imposer de posture à l'utilisateur :
  - possibilité de travailler assis ou debout.
  - possibilité de régler le plateau en position assise (20 cm au-dessus du bureau, la plate-forme ne doit pas empêcher la visualisation de l'écran).
  - possibilité de régler le plateau en position basse (20 cm) ou haute (entre 1 m et 1,10 m).
- en position de travail, la manipulation des interacteurs ne doit pas induire de fatigue :
  - possibilité de poser les avant-bras en position assise (barre d'avant-bras)
  - possibilité de ne pas poser les avant-bras en position debout (barre d'avant-bras amovible/escamotable).
  - délimitation de la zone d'acquisition de manière à permettre à l'utilisateur de ne pas lever les avant-bras pendant les manipulations (dimension maximale : 30cm / 45cm).

De plus, le système doit être transportable et ne pas être encombrant dans un bureau.

3.4.3 Conception détaillée et re-conception

Les étudiants nous ont proposé plusieurs solutions (cf. fig 3.29). Nous avions retenu la solution 1 qui nous semblait correspondre au mieux aux cahier des charges fonctionnel.

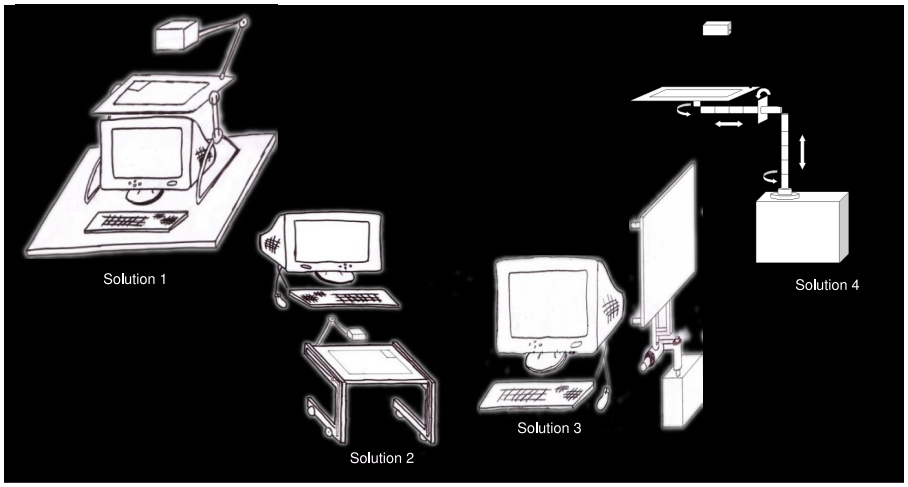


FIG. 3.29 – Croquis des solutions pour la plate-forme.

3.4.3.1 Première solution

À partir de ces croquis les étudiants ont réalisé des plans sous Catia[7] (cf photo. 3.30). La fabrication de la plate-forme était envisagée en métal.

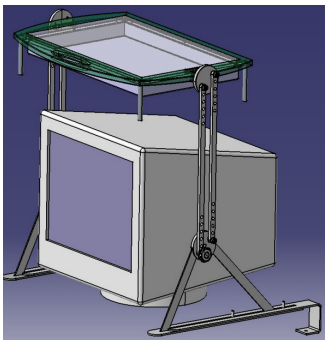


FIG. 3.30 – Modélisation sous Catia de la première solution choisie.

La quasi-totalité de la plate-forme devait être composée d'aluminium. Afin de concevoir les pièces dessinées, l'activité de l'entreprise devait être diversifiée (pliage, fraisage, tournage, poinçonnage). Il fut alors difficile de trouver des entreprises capables d'effectuer un tel usinage dans le cadre d'un prototype. La plupart des entreprises travaillent en moyenne et grande séries et sont spécialisées dans un seul domaine. Les entreprises ayant accepté de travailler sur un prototype nous

[7]Catia est un logiciel de CAO de l'éditeur Dassault System.

ont fourni des devis très élevés (supérieur à 10000 euros) qui ne répondaient pas aux contraintes économiques définies dans le cahier des charges.

3.4.3.2 Re-conception

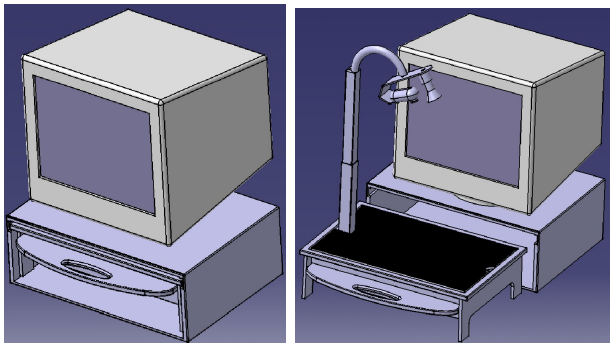


FIG. 3.31 – Version finale de la plate-forme.

Dans l'optique de l'évolution de cette solution onéreuse nous avons choisi un nouveau matériau : le plastique. En effet, il est facilement déformable, léger, suffisamment résistant et plus économique.
Pour la fabrication plastique, nous avons travaillé avec Mr Xavier Louvel responsable de la société APR2 (Applications plastiques régionales 2) localisée dans les Landes. Il a adapté la conception initiale aux contraintes du matériau en tenant compte du cahier des charges. Deux étudiants de l'ESTIA (Philippe Sain et Alexis Vachey) ont finalisé la conception de la nouvelle plate-forme.
Le principal changement de celle-ci est sa position dans l'espace de travail. Elle se range désormais dans un caisson en bois situé sous le moniteur et non plus au-dessus. La mobilité de la caméra par rapport au plateau est légèrement réduite (cf. fig. 3.31).

Chapitre 4

SKUA : Le prototype

Sommaire

SKUA (Système Kinésique Utilisable pour l'Assemblage) est l'application de nos travaux sur les interfaces tangibles et plus particulièrement sur la réalisation des interacteurs. Au cours de ce chapitre, nous exposons la concrétisation des travaux décrits précédemment. Dans la première partie nous présentons les interacteurs, dans la deuxième partie la plate-forme et dans la troisième partie un démonstrateur. Par la suite, nous décrivons une expérimentation menée auprès d'utilisateurs en vue de l'évaluation de SKUA. Une première expérience met en évidence la perception des interacteurs par l'utilisateur. Lors de la deuxième expérience, nous évaluons le démonstrateur et l'utilisation des interacteurs.

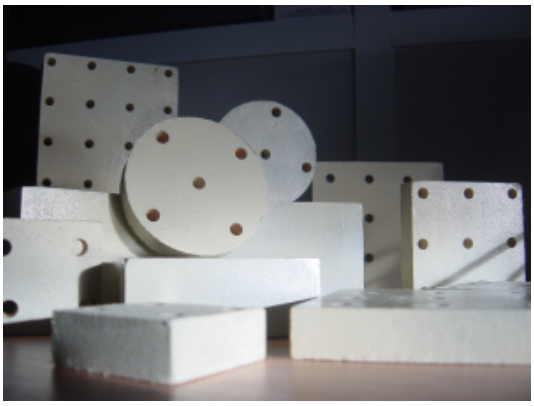


FIG. 4.1 – Photographie du premier jeu d'interacteurs.

4.1 Prototypage des interacteurs

À partir du travail de conception présenté dans la section 3.2.2, nous avons rédigé un cahier des charges [Gar03] et créé une modélisation de nos interacteurs en CAO avec le logiciel Catia. Dans le cahier des charges nous décrivons les contraintes d'utilisation des interacteurs en plus de leurs caractéristiques géométriques. Nous voulions un matériau léger, solide et agréable au toucher pour permettre une manipulation non fatigante. La matière ne devait pas subir d'usure importante et par conséquent ne pas s'effriter. En effet, une usure prononcée des perçages empêcherait les possibilités de fixation. Enfin, pour ne pas gêner le suivi vidéo cette matière ne devait pas être réfléchissante.

ESTIA-innovation, le centre de ressources technologiques de l'ESTIA, a réalisé la fabrication des interacteurs. La matière choisie est une résine polyuréthane d'une masse volumique de $0,65 g/cm^3$ (KUVO 15040). L'usinage de la forme extérieure a été réalisé avec un robot à commande numérique à 3 axes, les perçages ont été exécutés avec une perceuse à colonne. Aujourd'hui, SKUA dispose d'un jeu de soixante interacteurs répartis en deux formes (cylindre et parallélépipède) et trois tailles (cf. photo. 4.1). Après les perçages, les interacteurs ont été peints en blanc-cassée pour éviter que la matière ne s'effrite. Cette couche de peinture améliore également le toucher de l'interacteur et donc la qualité de manipulation. Le système de fixation utilisé est basé sur l'utilisation de goupillons de 5mm de diamètre. Pour maintenir deux interacteurs en position, nous insèrons un ou plusieurs goupillons dans les perçages du premier interacteur. Après avoir inséré les goupillons et donc créé des points de fixation insérés, nous devons joindre les interacteurs pour réaliser l'assemblage.

FIG. 4.2 – Photo-montage de la version finale de la plate-forme.

4.2 Prototypage de la plate-forme

La fabrication de la plate-forme a été réalisée par la société APR2 (située dans le département des landes 40), avec laquelle nous avions fait évoluer (cf. para. 3.4.3.2, page 84) notre conception vers une solution en plastique.
La plate-forme est principalement constituée de plastique pour des raisons de légèreté, de coût et de flexibilité (cf. photo. 4.2) expliquées dans la partie 3.4.3.2, page 84. La caméra est fixée sur un flexible assez rigide pour ne pas réagir au moindre contact mais suffisament souple pour être orienté dans la position souhaitée. Ce flexible est attaché à un bras métallique permettant de le régler à différentes hauteurs. Pour ranger la plate-forme, un caisson en bois a été fabriqué par le lycée Louis de Foix de Bayonne. Le caisson a été conçu pour pouvoir supporter un moniteur de PC d'une taille de vingt et un pouces. Ainsi, la plate-forme peut être rangée sous le moniteur afin de réduire l'encombrement sur le bureau.

4.3 Démonstrateur

Nous avons développé une interface en GTK+[gtk04] pour effectuer la calibration de la caméra. Cette première interface permettait d'interagir sur les paramètres de calibration et des valeurs des filtres, ainsi que de déclencher le suivi et d'agir sur les paramètres de la caméra (luminosité, contraste, netteté, mode de traitement, cadence des images). Pour l'acquisition des données vidéo, nous avons utilisé le pilote PWCX [1]. Il permet de contrôler une dizaine de modèles de caméra USB de marques Philips et Logitech. Nous l'utilisons avec deux modèles de webcam, la Philips TouCam Pro 740 et la Logitech Quickam 4000.

[1] Site officiel : http ://www.smcc.demon.nl/webcam/.

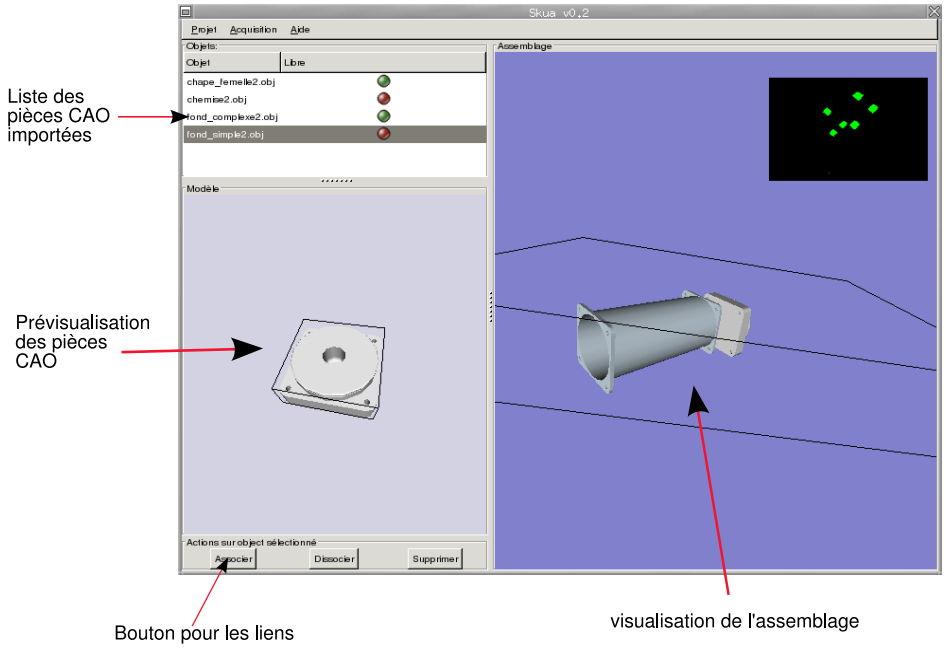


FIG. 4.3 – Capture d'écran de l'interface de SKUA.

De la même manière, pour le format géométrique nous avons utilisé le format "obj" de Alias WaveFront. Ce format est assez répendu dans les moteurs de rendu disponibles sur le web tels crystal space[Spaa] et openSG[Opeb]. La description des modèles est simple, car il n'y a pas de graphe de scène, et suffisante car nous utilisons uniquement la description géométrique des modèles CAO. De plus, le logiciel "ProEnginer" exporte des fichiers "obj" avec la possibilité de choisir la précision du maillage 3D ; dans une certaine mesure nous pouvons donc agir sur la complexité géométrique de nos modèles CAO.
En coopération avec Julien Hadim, étudiant en Master Recherche de l'université de Bordeaux 1, nous avons fait évoluer le programme vers un démonstrateur plus abouti.
Le logiciel (fig. 4.3) présente trois principaux cadres. Le premier affiche une prévisualisation 3D de la pièce CAO sélectionnée. Le second cadre affiche la liste des pièces CAO importées pour la session de travail en cours, elle permet la sélection des pièces CAO. Le troisième cadre est la vue 3D principale des pièces CAO qui sont associées à un interacteur et donc manipulées ; le point de vue est modifiable à l'aide de la souris tout comme pour la prévisualisation. Nous appelons l'espace affiché : la scène.
Avec l'interface logicielle, l'utilisateur peut créer un projet qui contient les informations des pièces importées. Les pièces CAO importées apparaissent dans la liste avec un icône indiquant si elles sont associées ou non à un interacteur. Pour

réaliser une association entre un interacteur et une pièce CAO, l'utilisateur dispose un interacteur sur le plateau. Les marqueurs (sous formes de petites pastilles vertes) permettant le suivi vidéo des interacteurs (cf. 3.3.1, page 54) doivent dans le champ de vision de la caméra. L'utilisateur sélectionne ensuite sa pièce CAO dans la liste, donne l'orientation dans la prévisualisation et valide l'association avec le bouton "*associer*". À partir de là, toutes les actions de déplacement dans le plan avec l'interacteur sont reproduites sur la pièces CAO dans la vue principale. L'utilisateur peut alors manipuler les interacteurs pour déplacer les pièces CAO dans la scène. À tout momment, il peut retirer une pièce CAO de la scène et annuler l'association entre l'interacteur et la pièce CAO en actionnant le bouton "*dissocier*".

4.4 Évaluations des interacteurs

Afin de valider l'approche théorique qui nous a permis d'établir notre premier jeu d'interacteurs, nous avons mené une expérience ([LGC04]) auprès de différents utilisateurs. Celle-ci consistait à leur donner la représentation 3D d'une pièce mécanique et à leur demander de réaliser cette pièce avec le jeu d'interacteurs mis à leur disposition.

4.4.1 Protocole expérimental

Notre objectif est de vérifier certaines hypothèses que nous avons émises lors de la conception de nos interacteurs. Nous voulons savoir si l'utilisateur se satisfait des formes existantes des interacteurs pour symboliser des pièces CAO, si les perçages sont suffisants pour les mises en position, si le choix de l'interacteur est lié à la forme de la pièce CAO. Nous cherchons également à connaître le pressentiment de l'utilisateur vis à vis des interacteurs. Nous définissons le pressentiment comme la connaissance intuitive de l'utilisation des interacteurs dans le cadre de la manipulation. L'échantillonnage de test est composé de quatre personnes : un concepteur, un expert assemblage (technologue), un utilisateur de CAO et un ergonome.

Lors de l'expérience, le sujet visualisait et manipulait séquentiellement dix pièces CAO sur l'écran d'un ordinateur. Il devait associer un interacteur à chaque pièce CAO et nous communiquer ses impressions, ses doutes et ses agacements. Certaines pièces CAO pouvaient l'amener à se demander s'il pouvait combiner les interacteurs pour créer un "nouvel interacteur" de forme plus proche de la pièce CAO. Pour la conduite de l'expérimentation, nous avons fourni au sujet un protocole détaillant le contexte, la finalité et le déroulement de l'expérience. Aucune indication n'a été donnée par rapport à l'utilisation des interacteurs (nombre d'interacteurs par pièce, forme ou taille identiques à la pièce CAO).
À la fin de l'expérience, nous avons recueilli les impressions. La saisie des informations a été réalisée avec une caméra vidéo.

FIG. 4.4 – Résultats de l'expérimentation du premier jeu d'interacteurs.

FIG. 4.5 – Exemple d'expérimentation.

4.4.2 Analyse de l'expérimentation, synthèse

Les résultats de cette expérimentation (cf. fig. 4.4) ont montré que la perception de l'objet initial différait selon les acquis et le conditionnement psychologique (c-à-d. le vécu) des utilisateurs. Certains sujets semblent utiliser le critère de forme et essaient de subdiviser la pièce de CAO avec des formes élémentaires. Dans ce cas, la forme générale de la combinaison des interacteurs est souvent un modèle simplifié de la pièce CAO. La majeure partie du temps, la forme générale des pièces CAO est relativement préservée dans les différents modèles établis avec les interacteurs. Les sujets plus spécialisés ont une approche plus technique : ils identifient les surfaces fonctionnelles de la pièce CAO et plus particulièrement celles prépondérantes dans les opérations d'assemblage.

Nous illustrons cette idée avec l'exemple du schéma 4.5, p.91. La pièce CAO, schéma de gauche, peut être définie comme la combinaison de deux formes élémentaires. Les surfaces fonctionnelles de la présente pièce sont (S1) la surface plane composée de quatre trous et la surface cylindrique (S2). La première proposition (figure centrale) est une combinaison d'interacteurs pour obtenir une forme générale proche de celle de la pièce CAO. Un interacteur est ajouté dans la seconde proposition (figure de droite) par rapport au cas précédent. Cette proposition a pour dessein de rendre effective (utilisable) la deuxième surface fonctionnelle (S2) avec les interacteurs.

L'expérience a donc mis en évidence deux raisonnements distincts pour l'utilisation des interacteurs. Dans le premier cas, le sujet cherche à réaliser une représentation de la forme géométrique de la pièce CAO avec les interacteurs. Dans le second cas, il considère les surfaces de contact entre les pièces et propose une combinaison d'interacteurs logique avec les fonctionnalités de l'assemblage.

Lorsque le sujet crée une représentation de la forme géométrique, il peut omettre des possibilités d'assemblage : par exemple, l'absence d'un interacteur pour représenter la surface (S2) (cf. fig. 4.5, p.91 en haut) enlève la possibilité de centrage dans l'assemblage. Cette représentation peut être un frein à l'assemblage. Les propositions basées sur l'analyse des surfaces fonctionnelles (deuxième raisonnement) sont plus pertinentes. Les fonctions d'assemblage entre les interacteurs et la pièce CAO sont proches. De plus, en s'appuyant sur ce raisonnement, la forme

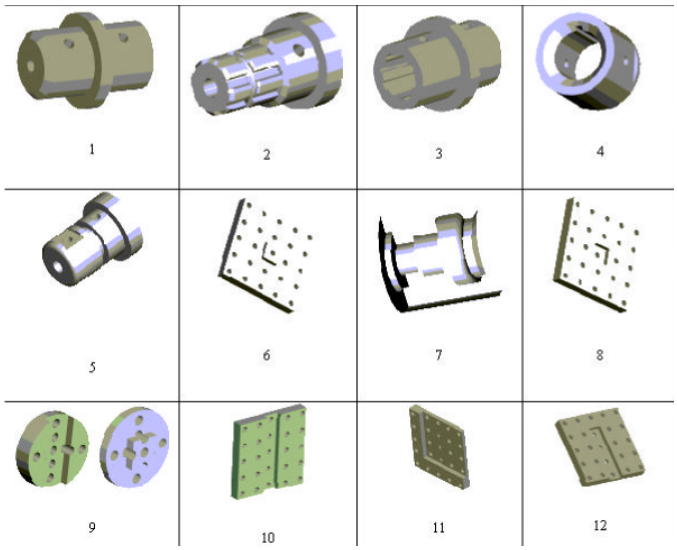


FIG. 4.6 – Exemple de surfaces fonctionnelles.

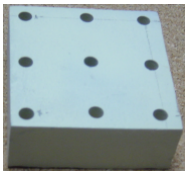
du modèle établi avec des interacteurs peut être relativement différente de la pièce CAO, ainsi le nombre de forme et de taille d'interacteurs peut être réduit.
Cette expérience nous a amené à faire évoluer le jeu d'interacteurs initial.

4.4.3 Évolution possible des interacteurs : les greffes

Un certain nombre de principes et d'éléments technologiques sont employés pour faciliter l'assemblage de pièces. Cependant, d'un point de vue mécanique, l'assemblage de pièces peut être décrit en deux étapes principales : le positionnement respectif des pièces et la fixation mutuelle des pièces. Un expert en assemblage cherche à identifier les surfaces fonctionnelles afin d'analyser ou optimiser l'ensemble du produit. Par conséquent, nous voulons faire évoluer notre premier jeu d'interacteurs basé sur les méthodes DFA vers de nouveaux interacteurs basés sur l'identification des surfaces fonctionnelles.

Nous réfléchissons actuellement à la mise au point d'un jeu plus complet prenant en compte la plupart des liaisons fonctionnelles comme le lamage (n°7 dans fig. 4.6) ou la mortaise (n°8 dans fig. 4.6). Le tableau de la figure 4.6, p 92 décrit des surfaces fonctionnelles que nous pouvons modéliser avec de nouveaux interacteurs. La photographie (cf. photo. 4.7) illustre les idées d'évolution que nous avons prévues pour les interacteurs. L'image 4.7 est un interacteur issu du premier jeu où nous avions mis en avant les mises en position en nous appuyant sur nos précédents travaux ([GLC03], [LGC03]) basés sur les méthodes DFA. En bas à droite (cf. photo. 4.7), une ébauche d'un interacteur basé sur les surfaces fonctionnelles. La photographie montre un interacteur avec une liaison glissière.

interacteur actuel



surfaces fonctionnelles

FIG. 4.7 – Évolution des interacteurs.

La création d'interacteurs basés sur les surfaces fonctionnelles implique la fabrication de nouveaux interacteurs. La diversité des surfaces fonctionnelles multipliée par les trois tailles possibles génèrerait un nombre d'interacteurs important. Trois inconvénients majeurs apparaitraient : l'encombrement, le coût de fabrication, la complexité d'utilisation. En effet, lors de l'association entre un interacteur et la pièce CAO, le nombre de possibilités peut gêner l'utilisateur dans son choix. Pour palier à ces problèmes, nous avons eu l'idée d'utiliser des "*greffes*". Nous employons ce terme car il s'agit de créer des petites plaques disposant d'une surface fonctionnelle. La dimension de la plaque et ces perçages nous permettent de l'adapter à tous les interacteurs, quelque soit leur taille. Sur les interacteurs de grande taille, la greffe peut être disposée à plusieurs endroits de sa surface, ce qui offre une possibilité de mise en position supplémentaire. L'idée des greffes est illustrée par les modèles CAO de la figure 4.8. À partir de ces modèles CAO nous avons réalisé des prototypes en bois visibles sur la photographie 4.9.

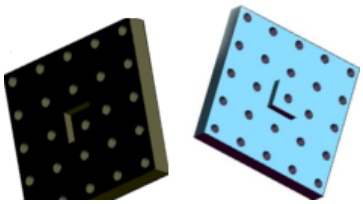


FIG. 4.8 – Modèle CAO des greffes.

Cette technique nous permet de conserver le premier jeu d'interacteurs que nous enrichissons par des surfaces fonctionnelles. L'évolution de nos interacteurs

devient plus simple à mettre en place, économique et moins limitée par le nombre de surfaces fonctionnelles. Nous pensons créer seulement deux greffes pour chaque surface fonctionnelle, car nous utilisons rarement plus de trois interacteurs simultanément.

Fabrice Depaulis, post-doctorant Informatique au LIPSI, réalise actuellement une étude exhaustive des éléments mécaniques de base utilisés dans les tâches d'assemblage les plus courantes. Cette étude servira de base lors de la conception des différentes surfaces fonctionnelles sous formes de greffes.

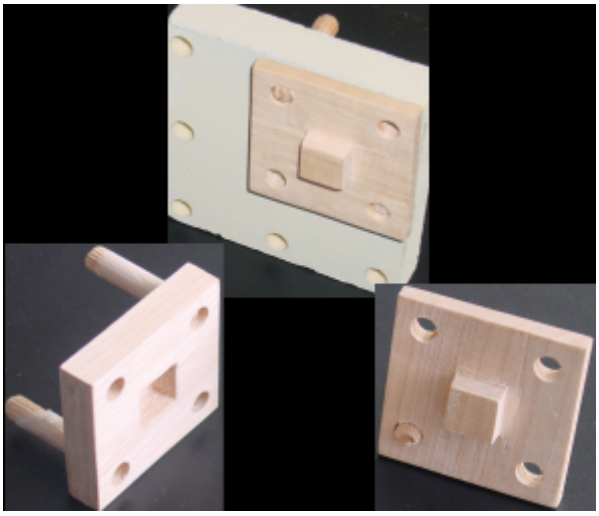


FIG. 4.9 – Prototypes des greffes.

4.5 Utilisation de SKUA

SKUA est la composition de l'interface logicielle, des interacteurs et de la plate-forme. Nous avons cherché à obtenir une première évaluation de l'ensemble des éléments de SKUA dans le cas d'une utilisation pratique. Dans la première partie, nous décrivons son installation dans un environnement de travail. Dans la deuxième partie nous présentons et commentons son utilisation.

4.5.1 Intégration en bureau d'étude

Un des objectifs de cette thèse était de créer une interface tangible, SKUA, pour la manipulation et l'assemblage de pièces CAO en respectant les contraintes suivantes. SKUA doit permettre la manipulation d'objets CAO par le biais d'objets réels, être d'un coût de fabrication équivalent à celui d'un ordinateur standard et s'intégrer sur un bureau sans modifier l'environnement. Notre intention était

de répondre aux contraintes liées à l'utilisation de SKUA dans un bureau d'étude. Pour répondre à cette fonctionnalité, nous avons travaillé avec Stéphanie Minel, post-doctorante, spécialiste de l'ergonomie. Après s'être renseignée sur l'origine, le contexte et l'historique du projet, elle nous aidé à dresser un questionnaire ([Min04]) sur la prise en compte des contraintes liées au bureau d'étude (espace et coût), à l'utilisateur (confort et apprentissage) et au travail (apport dans l'activité et modification de la manière de travailler). Dans la suite nous détaillons les réponses et les solutions que nous apportons à ces différentes questions.

Pour permettre une utilisation en bureau d'étude de PME, nous avons cherché à réaliser une solution économique pouvant se ranger facilement. Le coût de la plate-forme (inférieur à 1500 euros) offre une solution mono-poste même pour un petit bureau d'étude. L'utilisation du caisson en bois et la conception de la plate-forme répondent aux critères de place et de rangement. Lors de l'utilisation, la plate-forme peut être située au dessus du clavier et permet ainsi une posture cervicale adéquate sans modifier l'environnement de travail. Rangée, elle prend place sous le moniteur et dégage ainsi de l'espace sur le bureau. Actuellement, la taille de la plate-forme est jugée un peu importante et génère un encombrement conséquent, mais nous pensons pouvoir la réduire dans une seconde version.

4.5.2 Manipulation d'objets avec SKUA

L'utilisateur doit pouvoir manipuler avec les interacteurs les pièces 3D conçues à partir de logiciels de CAO. Pour faire le lien entre les interacteurs et les pièces CAO importées, l'utilisateur se sert de la souris et du clavier. Il sélectionne la pièce CAO à l'aide de la souris dans la liste et appuie sur le bouton "*associer*". Le prochain interacteur posé sur le plateau sera lié avec cette pièce CAO. Alors, les manipulations (déplacement, rotation) des pièces CAO se réalisent par le biais des interacteurs. Lors de la manipulation des interacteurs, l'utilisateur voit en permanence, à l'écran, les pièces CAO manipulées. Les autres informations sont générées par la position des interacteurs situés sur le plateau.

4.5.2.1 Description de la manipulation

Lorsqu'une association entre un interacteur et une pièce CAO est créée, toutes les actions sur l'interacteur sont reportées sur le modèle CAO.

Les interacteurs (cf. fig 4.10) permettent une interaction semi-directe pour le déplacement des objets et pour les actions d'assemblage. En effet, les déplacements des interacteurs suivant les trois degrés de liberté t_x, t_y, r_z sont reproduits à l'identique sur les pièces CAO. La détection des mouvements est réalisée à l'aide d'un système de caméra. L'utilisation d'une caméra permet d'éviter des contraintes tels les fils (centrale inertielle) ou les emboîtements (système électro-

nique). En revanche, l'utilisateur doit veiller à ne pas masquer les marqueurs de manière prolongée. L'utilisateur peut à tout moment ajouter, supprimer une pièce ou déplacer deux pièces en même temps.

	Nombre	**Description**
Mobilité(s) interprétée(s)	3	t_x, t_y, r_z
Contexte(s)	$= 0$	

FIG. 4.10 – Description de SKUA.

Lorsqu'il veut réaliser une action extérieure à la manipulation (sauvegarde, importation d'objet) il doit utiliser le clavier et la souris pour interagir avec le logiciel.

4.5.2.2 Constat

L'utilisation des interacteurs pour la manipulation des objets CAO est un atout. La manipulation est agréable et naturelle. Les personnes du laboratoire et quelques étudiants ayant manipulé les interacteurs ont rapidement réalisé les déplacements qu'elles souhaitaient sans commettre d'erreur. De plus, nous n'avons pas eu besoin d'expliquer le fonctionnement pour les déplacements.

Nous avons noté deux modalités à optimiser dans notre interface tangible, SKUA. La première concerne l'utilisation de la souris et du clavier que nous jugeons encore trop présente. La sélection des pièces CAO est réalisée avec la souris tout comme l'action de lier un interacteur à une pièce CAO. La seconde modalité concerne la perte de données propre aux déplacements : les interacteurs offrent six mobilités mais seules trois sont pour le momment prises en compte.

4.5.2.3 Évolution

Dans la taxinomie établie en 2.3.4, nous avons mis en avant le fait qu'une I2HD doit proposer une interaction semi-directe ou directe pour les actions principales, et une interaction symbolique pour les actions secondaires. Nous voulons donc modifier notre interface tangible afin de réduire l'utilisation du clavier et de la souris. En effet, l'utilisation de SKUA met en avant la répétition de plusieurs opérations faisant appel à la souris et au clavier, comme par exemple, l'association entre une pièce et un interacteur. La principale raison est que nous avons conçu nos interacteurs pour l'assemblage et la manipulation géométrique, les autres actions doivent donc être réalisées à l'aide du clavier et de la souris. De la même manière que nous avons conçu des interacteurs pour l'assemblage, nous voulons concevoir un interacteur pour accéder au fonctionnalités de l'interface (associer, dissocier). Pour cela, nous voulons détailler chaque cas d'utilisation pour déterminer les actions les plus souvent réalisées et celles contraignantes pour l'utilisateur.

Chapitre 5

Conclusion

Sommaire

Au cours de cette thèse, nous nous sommes intéressés à la réalisation d'une interface tangible adaptée aux contraintes d'assemblage et plus particulièrement à la réalisation de sa partie réelle. Nous avons exprimé les caractéristiques d'une interface tangible (chapitre 2) puis dégagé des principes de conception (chapitre 3) avant de traiter la réalisation (chapitre 4, 5, 6). Nous rappelons dans la conclusion de ce mémoire les principaux apports de notre travail, puis nous présentons les évolutions que nous envisageons.

5.1 Contributions

L'étude de l'interaction au travers du filtre de l'interaction homme-machine nous a permis de mettre en évidence l'interdépendance Utilisateur-Tâche-Application conduisant à la proposition de cinq règles de conception pour une interface utilisateur

Règle 1 *L'interacteur doit suggérer sa fonction par ses propriétés physiques.*

Règle 2 *L'espace d'action doit être confondu avec l'espace de perception.*

Règle 3 *L'interacteur doit avoir une forme symboliquement proche de celle de l'objet numériquement manipulé.*

Règle 4 *L'interacteur doit être tangible et manipulable sans effort directement avec les mains.*

Règle 5 *L'interacteur doit permettre d'accéder aux actions importantes de l'application de manière directe.*

Au delà de ces règles de conception spécifiques aux interfaces tangibles, nous avons établi une taxinomie sur les principes d'interaction, d'après laquelle nous avons fait émerger trois principes d'interaction : l'interaction symbolique, l'interaction semi-directe et l'interaction directe.
Nous avons défini la manière de choisir le type d'interaction en fonction de l'utilité et de l'usage de l'action : l'interaction symbolique pour les actions secondaires ou difficiles à appréhender et les interactions directe ou semi-directe pour les actions fréquentes ou notables.

La première version du démonstrateur SKUA permet de joindre et de manipuler plusieurs pièces CAO simultanément à l'aide des interacteurs. Il nous a permis de progresser dans notre approche de l'interaction et plus particulièrement des interfaces tangibles.
L'interface tangible SKUA démontre que l'utilisation des interacteurs pour manipuler les objets dans l'espace est vraiment pertinente et performante. La manipulation des objets et l'association avec les pièces 3D sont naturelles pour les utilisateurs. Nous approuvons les travaux de Ware et Rose ([WR99]) en remarquant que les différences de géométrie et de tailles entre les interacteurs et les pièces CAO ne sont pas un frein pour la manipulation.
L'utilisateur associe intuitivement les interacteurs aux pièces CAO car le positionnement relatif des interacteurs dans l'espace réel est identique au positionnement des pièces CAO dans l'espace virtuel. SKUA nous a donc permis de mettre en avant l'importance de la disposition spatiale des interacteurs par rapport à leurs formes et leurs tailles.
En revanche, nous avons montré que la différence de taille entre les objets réels et virtuels pose des problèmes pour la tâche d'assemblage. En effet, deux interacteurs en contact n'impliquent pas que les pièces CAO associées soient en contact et inversement, des pièces CAO en contact n'impliquent pas que les interacteurs soient en contact. Nous cherchons une solution logicielle à ce problème car nous ne pouvons pas reproduire tous les rapports de taille existant entre les pièces CAO et les interacteurs.
Nous nous sommes aperçu également que les utilisateurs étaient gênés par la possibilité de mouvoir les interacteurs suivant les six degrés de liberté alors que le modèle CAO est contraint dans le plan.
Nous avons également montré les limitations des interfaces tangibles utilisant des interacteurs conçus par rapport aux méthodes DFA en pointant l'absence de surface fonctionnelle.

5.2 Perspectives

Une partie de cette thèse a été consacrée à la réalisation de la partie tangible de notre interface ([GC03],[GLC03],[LGC03]) et à son évolution ([LGC04]). Ces recherches étant avancées, nous avons deux principales perspectives de travail : une évolution de la partie logicielle et la réalisation d'un suivi des six degrés de liberté de nos interacteurs.

Nous cherchons une solution logicielle au problème lié à la différence de taille entre les objets réels et virtuels pour la tâche d'assemblage : nous ne pouvons pas reproduire tous les rapports de taille existant entre les pièces CAO avec les interacteurs. Une évolution possible serait de reconnaître les collisions et les surfaces de contact entre deux pièces CAO à partir des modèles géométriques.
Notre but est de permettre un assemblage plus précis. Pour celà, nous devons détecter les collisions des faces entre chaque objet de manière suffisamment précise. En connaissant la position des faces en contact, nous pouvons calculer l'orientation des pièces pour permettre les assemblages face à face. Actuellement, nous testons les différentes bibliothèques du UNC Gamma Research Group : Geometric Algorithms for Modeling, Motion and Animation ([oCSUoNC04]) tels PQP [GLM96], SWIFT++ [EL01] et I-COLLIDE ([GRLM03]). À partir de ces évaluations nous cherchons le meilleur compromis entre la performance en temps de calcul, la précision des tests et la simplicité d'intégration dans notre application. Notre modèle géométrique devenant plus complexe, nous voulons mettre en œuvre une structure de stockage des informations, plus évoluée, sous forme de graphes. En effet les structures de graphes sont un élément commun aux domaines de l'informatique et de l'assemblage. En informatique, il existe beaucoup d'algorithmes s'appuyant sur les graphes et ces structures sont parfaitement connues. En assemblage ([Rej00]), les graphes sont utilisés pour représenter les assemblages : les nœuds sont les sous-parties de la pièce et les arêtes sont les types de liaisons. Nous voulons confondre les deux approches pour faire émerger un système de fichier au format XML pour les assemblages de pièces CAO.

Notre second objectif est l'évolution vers la reconnaissance des six degrés de liberté. Nous voulons permettre à l'utilisateur de manipuler les interacteurs dans un espace en trois dimensions plus vaste pour offrir plus de liberté dans la manipulation.
L'utilisation d'une webcam nous a permis de réduire fortement le coût de notre interface tangible et de posséder rapidement une version utilisable. En revanche, cette solution implique des restrictions importantes sur la qualité des couleurs, la taille de la surface de travail et plus globalement la collecte des informations sur le plateau (position et orientation des interacteurs, lignes, points, marqueurs). Nous avons acquis durant le dernier mois de cette thèse une caméra de meilleure qualité avec un angle d'acquisition plus important. Nous allons développer la liaison entre

la caméra et SKUA puis affiner les algorithmes du suivi par modèle pour réaliser un suivi 3D.

En 2001, au début de ce travail, nous avions délibérément écarté une solution à base de capteurs de positions et d'orientations. En effet, ces capteurs sont aujourd'hui assez coûteux, de l'ordre de trois milles euros, volumineux, de l'ordre de 20 cm3 et intrusifs car tous munis de fils. Ces trois points vont à l'encontre des contraintes du projet initial (présentées page 3 de ce mémoire). Cependant, la veille que nous avons effectuée ces trois dernières années sur les capteurs montre qu'ils suivent une rapide évolution tant sur leur taille qui est en nette diminution que sur leur autonomie pour laquelle de nombreux travaux de recherche sont menés. Cette veille nous amène à penser que les capteurs sont une solution efficace et prometteuse pour le suivi en 3D. Aussi, c'est cette voie que nous allons explorer en priorité dans l'avenir. A cet effet, nous venons d'acquérir des capteurs liberty 240-4 system de chez Polhemus.

Bibliographie

[3Dn02] 2002. Disponible sur World Wild Web [référence du 7 janvier 2002], http ://www.sgi.com/fun/freeware/3d_navigator.html.

[AFM+00] D. Anderson, J. Frankel, J. Marks, A. Agarwala, P. Beardsley, J. Hodgins, D. Leigh, K. Ryall, E. Sullivan, and J. Yedida. Tangible interaction and graphical interpretation : A new approach to 3d modeling. ACM siggraph, ISBN 1-58113-208-5, pages 393–402, 2000.

[Ais79] R. Aish. 3d input for caad systems. Computer-Aided Design, 11(2), 1979.

[ASZ+01] P. Abolmaesumi, S.E. Salcudean, W.H. Zhu, S.P. DiMaio, and M.R. Sirouspour. A user interface for robot-assisted diagnostic ultrasound. In IEEE Int. Conf. on Rob. and Auto. (ICRA), Korea, pages 1549–1554, May 2001.

[BBKf97] R. Balakrishnan, T. Baudel, G. Kurtenbach, and G. fitzmaurice. The rockin'mouse : Integral 3d manipulation on a plane. Procceding of 1997 ACM Conference on Human Factors in Computing Systems, pages 311–318, 1997.

[BD83] Boothroyd and Dewhurst. Design for assembly - a designer's handbook, 1983.

[BDC00] M. Brown, T. Drummond, and R. Cipolla. 3d model acquisition by tracking 2d wireframes. In BMVC00, 2000.

[BDW+03] Aaron Bloomfield, Yu Deng, Jeff Wampler, Pascale Rondot Dina Harth, MAry McManus, and Norman Badler. A taxonomy and comparison of haptic actions for disassembly tasks. In Proceedings of the 2003 IEEE Virtual Reality Conference (VR'03), pages 225–231, March 2003.

[BGF00] William Buxton and Gordon Kurtenbach George Fitzmaurice, Ravin Balakrishnan. Large displays in automotive desigin. IEEE Computer Graphics, 20(4) :pages 68–75, July/August 2000.

[BH97] Peter Berkelman and Ralph Hollis. Dynamic performance of a magnetic levitation haptic device. In Proceedings of SPIE Telemanipulation & Telepresence, 1997.

[BH00] Y. Boykov and D.P. Huttenlocher. Adaptive bayesian recognition in tracking rigid objects. In CVPR00, pages 697–704 (II), 2000.

[Bru98] F. W. Bruns. Integrated real and virtual prototyping. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON 98), Aachen, Germany.IEEE., 4 :pages 2137–2142, 1998.

[Bux02] B. Buxton, 2002. Disponible sur World Wild Web [référence du 6 avril 2002],
http ://www.billbuxton.com/InputSources.html.

[Can86] J. Canny. A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6) :pages 679–698, 1986.

[Car97] John M. Carroll. Human-computer interaction : psycholopy as a science of design. Human-Computer Studies, pages 501–522, 1997.

[CC99] A. Cohen and E. Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In In Proceedings of the ASME Winter Annual Meeting, Dynamics Systems and Control, pages 401–402, 1999.

[CCG+04] A. Candeau, F. Combellas, P. G., B. Lajusand, and N. Rey. Eskua, réalisation d'une plateforme. Projet de Fin d'étude ingénieur ESTIA, 2004.

[CFH97] L. D. Cutler, B. Frölich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In Proceedings of the 1997 symposium on Interactive 3D graphics, pages pages 107–ff. ACM Press, 1997.

[Che99a] E. Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In Proceedings of the First International Workshop on Virtual Reality and Prototyping, pages pages 97–106, juin 1999.

[Che99b] E. Chen. Six degree-of-freedomhaptic system for desktopvirtual prototyping applications, 1999.

[CMN83] S.K Card, T.P Moran, and A. Newell. The psychology of human computer interaction. HILLsdale, NJ :Lawrence Erlbaun Associates, 1983.

[CMS88] M. Chen, S. J. Mountford, and A. Sellen. A study in interactive 3-d rotation using 2-d control devices. Computer Graphics, 22(4) :pages 121–129, 1988.

[CNS93] J. Coutaz, L. Nigay, and D. Salber. Taxonomic Issues for Multimodal and Multimedia Interactive Systems. In Proceedings of the ERCIM Workshop on Multimodal HCI, pages 3–11, November 1993.

[Cor03] Intel Corporation. Open source computer vision library, reference manual. Technical Report Order number 123456-001, 2003. http ://prdownloads.sourceforge.net/opencvlibrary/OpenCVReferenceManual.pdf.

[CSL] Sony CSL : . Disponible sur World Wild Web [référence du 13 janvier 2002]
http ://www.csl.sony.co.jp/person/rekimoto/toolstone/.

[DC00] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In ECCV (2), pages 20–36, 2000.

[DCGL05] Fabrice Depaulis, Nadine Couture, Ludovic Garreau, and Jérémy Legardeur. A reusable methodology based on filters in order to define relevant tangible parts for a tui. In Engineering Reality of Virtual Reality 2005. part of IS&T/SPIE Symposium on Electronic Imaging 2005, San Jose (Californie, USA), 2005.

[dlRG03] J.B. de la Rivière and P. Guitton. Hand posture recognition in large display VR environments. In Gesture Workshop, April 2003.

[EL01] Stephan A. Ehmann and Ming C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In A. Chalmers and T.-M. Rhyne, editors, EG 2001 Proceedings, volume 20(3), pages 500–510. Blackwell Publishing, 2001.

[ERC] News No.37 (April 1999) ERCIM : . Disponible sur World Wild Web [référence du 16 juin 2002]
http ://www.ercim.org/publication/Ercim_News/enw37/goebel.html.

[ESB99] H. Ernst, K. Schäfer, and W. Bruns. Creating virtual worlds with a graspable user interface. Interactions in Virtual Worlds, Proceeding of the Twente Workshop on Language Technology, TWLT 15, Enscheide : Universiteit Twente, S, pages 45–57, 1999.

[FBR86] M. Fjeld, M. Bichsel, and M. Rauterberg. Build-it : a brick-based tool for direct interaction, 1986.

[FIB95] George W. Fitzmaurice, Hiroshi Ishii, and William Buxton. Bricks : Laying the foundations for graspable user interfaces. In Computer Human Interaction (CHI), pages 442–449, 1995.

[FMP01] P. Fuchs, G. Moreau, and J.P. Papin. Le traité de la réalité virtuelle. ISBN : 2-911762-34-7. Les Presses de l'École des Mines, novembre 2001.

[FPWa00] B. Frölich, J. Plate, J. Wind, and al. Cubic-mouse-based interactaction in virtual environments. IEEE Computer Graphics and Applications, pages 12–15, juillet aout 2000.

[FSSK02] M. Fjeld, S. Guttormsen Schär, D. Signorello, and H. Krueger. Alternative tools for tangible interaction : A usability evaluation.

In In Proceedings of IEEE and ACM International Symposium of Mixed and Augmented Reality (ISMAR 2002), pages 157–166, 2002.

[Gan] Disponible sur World Wild Web [référence du 2 octobre 2003] http ://gandalf-library.sourceforge.net/.

[Gar03] Ludovic Garreau. Cahier des charges des interacteurs d'eskua. disponible sur demande l.garreau@estia.fr, 2003.

[GC03] L. Garreau and N. Couture. Study of tangible user interface for handling tridimensionnal objects. In Proc. Real World User Interfaces, PI'2003, Udine (Italy), pages 64–68, September 2003.

[Geo] Geometry Technologies Geomview. Disponible sur World Wild Web [référence du 15 juillet 2002] http ://www.geomview.org/.

[GLC03] L. Garreau, J. Legardeur, and N. Couture. Tangible interface for mechanical CAD parts assembly. in the proceedings of Virtual Concept 2003,Editors D. Coutellier, X. Fischer, D. Marquis, I. Thouvenin, Biarritz 2003. ISBN 2-9514772-3-6, pages 222–227, novembre 2003.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree : A hierarchical structure for rapid interference detection. Computer Graphics, 30(Annual Conference Series) :pages 171–180, 1996.

[GMK99] P.J. Gorman, A.H. Meier, and T.M. Krummel. Simulation and virtual reality in surgical education - real or unreal. ARGH SURG, 134, november 1999.

[GOH98] Gorbet, M. Orth, and H.Ishii. Triangles : Tangible interface for manipulation and exploration of digital information topography. In ACM Press, editor, Proceedings of SIGCHI '98, pages 49–56, 1998.

[Gre97] S. Greenberg. A taxonomy of human computer interaction, 1997. http ://pages.cpsc.ucalgary.ca/~saul/hci_topics/papers/ ACMtaxonomyofHCI/taxonomy.html.

[GRLM03] N. Govindraju, S. Redon, M. Lin, and D. Manocha. Cullide : Interactive collision detection between complex models in large environments using graphics hardware. In In Proc. ACM SIGGRAPH/Eurographics Workshop Graphics Hardware, pages 25 – 32, 2003.

[Gro] Sato-Koike Group. Disponible sur World Wild Web [référence du 27 aout 2003] http ://sklab-www.pi.titech.ac.jp/frame_index-e.html.

[gtk04] gtk, 2004. Disponible sur World Wild Web [référence du 22 avril 2004] http ://www.gtk.org.

[HBC+96] Hewett, Baecker, Card, Carey, Gasen, Matei, Perlaman, Straong, and Verplank. Acm SIGCHI curricula for human-computer interaction. Technical report, ACM SIGCHI, 1996.

[HH93] H.R. Hartson and D. Hix. Developing User Interfaces : ensuring usability through product & process. ISBN : 0-471-57813-4. John Wiley & Sons, Inc. New York, NY,USA, 1993.

[HLGB03] J.M. Hasenfratz, M. Lapierre, J.D. Gascuel, and E. Boyer. Real-time capture, reconstruction and insertion into virtual world of human actors. In Vision, Video and Graphics 2003, 2003.

[HM95] R. Horaud and O. Monga. Vision par ordinateur : outils fondamentaux. Éditions Hermès, Paris, 1995.

[HPGK94] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Passive real-world interface props for neurosurgical visualization. In Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems, volume 1, pages 452–458, 1994.

[HS88] C. J. Harris and M. Stephens. A combined corner and edge detector. In In Proc. 4th Alvey Vision Conf., Manchester, pages 147–151, 1988.

[IS98] John Isidoro and Stan Sclaroff. Active voodoo dolls : A vision based input device for non-rigid control. In Proceedings of the Computer Animation, ISBN :0-8186-8541-7, June 1998.

[Kar02] R. Karlsson, 2002. Disponible sur World Wild Web [référence du 24 mars 2002],
http ://www.3dwm.org.

[KIK01] Y. Kitamura, Y. Itoh, and F. Kishino. Real time interaction with activecube. CHI 2001 Short Talks, March 31 - April 5, 2001, Seattle, Washington, USA., pages 355–356, 2001.

[KIMK00] Yoshifumi Kitamura, Yuichi Itoh, Toshihiro Masaki, and Fumio Kishino. Activecube : A bi-directional user interface using cubes. KES 2000 : Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, pages 99–102, 2000.

[KTY97] Kiyoshi Kiyokawa, Haruo Takemura, and Naokazu Yokoya. Manipulation aid for two-handed 3-d designing within a shared virtual environment. Proceedings of the 7th International Conference on Human-Computer Interaction jointly with 13th Symposium on Human Interface (HCI International '97), 1997.

[KTY00] K. Kiyokawa, H. Takemura, and N. Yokoya. SeamlessDesign for 3D object creation. IEEE MultiMedia, 7(1) :pages 22–33, January 2000.

[LCea] Anatole L'Ecuyer, Sabine Coquillart, and et al. A haptic prototype for the simulations of aeronautics mounting/unmounting operations.

[Leg98] J. Legardeur. Prise en compte des contraintes d'assemblage dans une logique de conception intégrée : l'approche design for assembly. Master's thesis, Université Joseph Fourier, Laboratoire 3S, Grenoble,, 1998.

[LGC03] J. Legardeur, L. Garreau, and N. Couture. Des interacteurs pour l'assemblage mécanique en CAO. 3rd International Conference : Integrated Design and Production, CPI'2003, Meknes, Maroc, 2003.

[LGC04] J. Legardeur, L. Garreau, and N. Couture. Experiments to evolve toward a tangible user interface for cad parts assembly. in Proc. Electronic Imaging Science and Technology, Stereoscopic Displays and Virtual Relality Systems XI, Editors : Andrew J. Woods, John O. Merrit, Stephen A. Benton, Mark T. Bolas, SPIE, EI'2004 SPIE, San Jose (US), 5291 :438–445, 18-22 january 2004.

[LMB+02] A. Lécuyer, C. Megard, J.M. Burkhardt, T. Lim, S. Coquillart, P. Coiffet, and L. Graux. The effect of haptic, visual and auditory feedback on an insertion task on a 2-screen workbench. IPT'2002 Symposium (Immersive Projection Technology),Orlando, US, 2002.

[LSR+00] B. Leibe, T. Starner, W. Ribarsky, Z. Wartell, D. Krum, J. Weeks, B. Singletary, and L. Hodges. The perceptive workbench : Toward spontaneous interaction in semi-immersive virtual environments. In Proceedings of IEEE Virtual Reality 2000, New Brunswick, NJ, pages 13–20, March 2000.

[LST01] Robert W. Lindeman, John L. Sibert, and James N. Templeman. The effect of 3d widget representation and simulated surface constraints on interaction in virtual environments. In VR, pages 141–148, 2001.

[LTSC02] R. Lindeman, J. Templeman, J. Sibert, and J. Cutler. Handling of virtual contact in immersive virtual environments : beyond visuals, 2002.

[Maya] 3dNY, GraphLink Studio. Disponible sur World Wild Web [référence du 11 juin 2002]
http ://www.3dny.org/news/maya.phtml.

[Mayb] Alias WaveFront Maya. Disponible sur World Wild Web [référence du 11 juin 2002]
http ://www.aliaswavefront.com/.

[MBL+04] Jun Murayama, Laroussi Bouguila, Y.L. Luo, Katuhito Akahane, Shoichi Hasegawa, B. Hirsbrunner, and Makoto Sato. Spidar g&g :

A two-handed haptic interface for bimanual vr interaction. In Eurohaptics 2004 Conference, Munich,, pages 138–146. Eurohaptics 2004 Conference, 2004.

[McL03] Philip F. McLauchlan. Gandalf : The fast computer vision and numerical library. Technical report, Imagineer Systems Ltd., 2003. http ://gandalf-library.sourceforge.net/tutorial/report.pdf.

[MF97] Wendy E. Mackay and Anne-Laure Fayard. Hci, natural science and design : Aframework for triangulation across disciplines. Designing Interactive Systems, pages 223–234, August 1997.

[Min04] Stephanie Minel. Questionnaire pour l'évaluation ergonomique de skua. disponible sur demande l.garreau@estia.fr, 2004.

[MP96] Rauterberg M and Steiger P. Pattern recognition as a key technology for the next generation of user interfaces. In Proc. of SMC'96, IEEE Catalog Number : 96CH35929, Vol. 4 :2805–2810, 1996.

[MRF+96] William R. Mark, Scott C. Randolph, Mark Finch, James M. Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems : Issues and solutions. Computer Graphics, 30(Annual Conference Series) :447–452, 1996.

[MS86] J. Wiley Mosier and J.N Smith. Application of guidelines for designing user interface software. Bahaviour and Information Technology, Vol. 5(No. 1) :pages 39–46, 1986.

[Mun95] A. S. Munroe. Is your design a life sentence ? Machine Design, page 156, January 1995.

[NC96] L. Nigay and J. Coutaz. Espaces conceptuels pour l'interaction multimédia et multimodale. TSI, spécial Multimédia et Collecticiel, 15(9) :pages 1195–1225, 1996.

[NHMA03] J. New, E. Hasanbelliu, M., and Aguilar. Facilitating user interaction with complex systems via hand gesture recognition. In In Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia., 2003.

[oCSUoNC04] Department of Computer Science University of North Carolina, 2004. Disponible sur World Wild Web [référence du 8 mai 2004] http ://www.cs.unc.edu/~geom/collide/packages.html.

[OLF] Office québécois de la langue française, Grand dictionnaire terminologique. Available on World Wild Web [may, the 16th 2003], *http ://www.granddictionnaire.com/*.

[Opea] Intel Corporation OpenCV. Disponible sur World Wild Web [référence du 28 juin 2002] http ://www.intel.com/research/mrl/research/opencv/.

[Opeb] OpenSG. Disponible sur World Wild Web [référence du 28 juin 2002] http ://www.opensg.org/.

[Pat99] G. Patry. Contribution à la conception du dialogue homme-machine dans les applications graphiques interactives de conception technique : le système GIPSE. PhD thesis, École Nationale Supérieure de Mécanique et d'Aérotec,et Faculté des Sciences Fondamentales et Appliquée, 1999.

[PHA] SensAble technologies : PHANTOM 1.5. Disponible sur World Wild Web [référence du 23 aout 2002]
http ://www.sensable.com/haptics/products/phantom.html.

[PPS99] J. Pierce, R. Pausch, and B. Stearns. Voodoo dolls : Seamless interaction at multiple scales in virtual environments. In Symposium on Interactive 3D Graphics, pages pages 141–145, 1999.

[R.79] Aish R. 3d input for caad systems, computer-aided design, 1979.

[RC94] A. Redford and J. Chal. Design for Assembly : Principles and Practice. ISBN : 0077078381. New York : Mc Grawn Hill, London, 1994.

[RCT] Reachin Reachin Core Technology. Disponible sur World Wild Web [référence du 23 aout 2002]
http ://www.reachin.se/businessareas/coretechnology/.

[Rej00] N. Rejnéri. Détermination et simulation des opérations d'assemblage lors de la conception de systèmes mécaniques. PhD thesis, Thèse INPG, 2000.

[RS00] J. Rekimoto and E. Sciammarella. Toolstone : Effective use of the physical manipulation vocabularies of input devices. Proc. of UIST 2000, pages 109–117, 2000.

[Sch91] C. Schepacz. Des méthodes pour un assemblage-montage rationnel. Cetim-Information n°123, Juillet 1991.

[SCP95] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM : Interactive worlds in miniature. In Proceedings CHI'95, pages pages 265–272, 1995.

[SDC04] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(4,) :pages 479–494, 2004.

[SHEB00] K. Schmudlach, E. Hornecker, H. Ernst., and F.W. Bruns. Bridging reality and virtuality in vocational training. CHI 2000 Extended Abstracts (Interactive Poster), pages 137–138, 2000.

[SI00] Sriram Subramanian and Wijnand IJsselsteijn. Survey and classification of spatial object manipulation techniques. In Proceedings of OZCHI 2000, Interfacing Reality in the New Millennium, December 2000.

[SIW+02] E. Sharlin, Y. Itoh, B. Watson, Y. Kitamura, L. Liu, and S. Sutphen. Cognitive cubes : A tangible user interface for cognitive assessment. In ACM CHI 2002 Conference Proceedings, pages 347–354. ACM Press New York, NY, USA, April 20-25 2002.

[SP97] S.E. Salcudean and N.R. Parker. 6-dof desk-top voice-coil joystick. 6th Annual Symposium on Haptic Interfaces for Virtual Environments and Teleoperation Systems, Intl. Mech. Eng. Congr. Exp., (ASME Winter Annual Meeting), 61 :131–138, Dallas, Texas, November 1997.

[Spaa] Crystal Space. Disponible sur World Wild Web [référence du 28 juin 2002]
http ://crystal.sourceforge.net/tikiwiki/tiki-view_articles.php.

[Spab] 3D Connexion SpaceMouse. Disponible sur World Wild Web [référence du 23 aout 2003]
http ://www.3dconnexion.com.

[SSWF00] S. Sutphen, E. Sharlin, B.A. Watson, and John Frazer. Reviving a tangible interface affording 3d spatial interaction. Proc. 11th Western Canadian Computer Graphics Symposium, (Panorama, Canada, March), pages 155–166, 2000.

[Sut63] I.E. Sutherland. Sketchpad : A man-machine graphical communication system. AFIPS Spring Joint Computer Conference, 1963.

[SWS+01] E. Sharlin, B.A. Watson, S. Sutphen, R. Lederer, P. Figueroa, and J. Frazer. 3d computer interaction using physical objects : exploration of tangible user interfaces. Leonardo Electronic Almanac, 9(7), 2001.

[Tan00] Hong Z. Tan. Creating interfaces that envelop a sense of touch has met with measuser success. Communication of ACM, 43 :pages 40–41, march 2000.

[TCH+03] N. Tarrin, S. Coquillart, S. Hasegawa, L. Bouguila, and M. Sato. The stringed haptic workbench : a new haptic workbench solution. Computer Graphics Forum, 22(3), 2003.

[The01] D. Thevenin. Adaptation en Interaction Homme-Machine : le cas de la plasticité. PhD thesis, Université Josehp Fourier - Grenoble 1, decembre 2001.

[Tic94] S. Tichkietwitch. De la cfao à la conception intégrée. Revue Internationale de C.F.A.O et d'Infographie, 9(5), 1994.

[TOBM02] A. Tang, C. Owen, F. Biocca, and W. Mou. Experimental evaluation of augmented reality in object assembly task. In In Proceedings of ISMAR '2002, IEEE and ACM International Symposium on Mixed and Augmented Reality, ISBN 0-7695-1781-1, pages 265–266. Darmstadt, Germany, October 2002.

[TOBM03] A. Tang, C. Owen, F. Biocca, and W. Mou. Comparative effectiveness of augmented reality in object assembly. In Cockton G Korhonen P. In Bellotti V, Erickson T, editor, Proceedings of ACM CHI'2003, Conference on Human Factors in Computing System, ISBN 1-58113-630-7, pages 73–80. Fort Lauderdale, FL. New York, NY : ACM Press., April 2003.

[UI97a] B. Ullmer and H. Ishii. The metadesk : Models and prototypes for tangible user interfaces. In Proceedings of UIST'97, pages 223–232, October 1997.

[UI97b] B. Ullmer and H. Ishii. Tangible bits : Towards seamless interfaces between people, bits and atoms. In ACM Press Atlanta, editor, Proceedings of the Conference on Human Factors in Computing Systems (CHI '97), pages 234–241, March 1997.

[UI00] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. IBM SYSTEMS JOURNAL, 39(3& 4) :pages 915–931, 2000.

[UI01] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. Human-Computer Interaction in the New Millennium, pages 579–601, 2001.

[UIG98] B. Ullmer, H. Ishii, and D. Glas. mediablocks : Physical containers, transports, and controls for online media. Proceedings of SIGGRAPH'98, ACM Press : New York Publ., pages 379–386, 1998.

[UIJ03] B. Ullmer, H. Ishii, and R. Jacob. Tangible query interfaces : Physically constrained tokens for manipulating database queries. In Proceedings of INTERACT'03 (September 2003), 2003.

[Ull95] B. A. Ullmer. Models and mechanisms for tangible user interfaces. Master's thesis, University of Illinois, 1995.

[Ull02] B. A. Ullmer. Tangible Interfaces for Manipulating Aggregates of Digital Information. PhD thesis, MIT Media Lab, 2002.

[UUI99] J. Underkoffler, B. Ullmer, and H. Ishii. Emancipated pixels : Real-word graphics in the luminous room. In ACM, editor, In : Proc. of SIGGRAPH'99. Los Angeles, CA, pages 385–392, August 1999.

[VS04] A. Vachey and P. Sain. Eskua, réalisation d'une plateforme. Rapport de Stage ingénieur ESTIA, 2004.

[Wik] Disponible sur World Wild Web [référence du 20 juin 2004], http ://fr.wikipedia.org/wiki/Accueil.

[WR99] C. Ware and J. Rose. Rotating virtual objects with real handles. ACM Transactions on CHI, 6(2) :pages 162–180, 1999.

[xse04] xsens, 2004. Disponible sur World Wild Web [référence du 22 avril 2004] http ://www.xsens.com/.